1. Problem 3-3 part a from the CLRS text. You need not provide justifications for your order.
   If you aren’t able to find a group, you can do this individually. While I encourage you to use LaTex, it is not strictly required for this assignment. You may hand write these solutions.
   Due Friday January 6.

2. Problem 3-4. For conjectures that are untrue, explain which reasonable class of functions the statement is true for. For example, you might say ”The statement is untrue in general, and here is an example. But the statement is true if the functions are strictly increasing, and here is why.
   Due Monday January 9.

3. Consider the following definitions for \( f(n, m) = O(g(n, m)) \). State which definitions are logically equivalent. Fully justify your answers. Argue about which definition you think is the best one in the context where \( f \) and \( g \) are run times of algorithms and the input size is monotonically growing in \( n \) and \( m \) (for example \( n \) might be the number of vertices in a graph and \( m \) might be the number of edges in a graph).

   (a) There exists positive constants \( c > 0, n_0, m_0 \) such that for all \((n, m)\) where \( n \geq n_0 \) and \( m \geq m_0 \) it is the case that \( f(n, m) \leq c g(n, m) \)

   (b) There exists positive constants \( c > 0, n_0, m_0 \) such that for all \((n, m)\) where \( n \geq n_0 \) or \( m \geq m_0 \) it is the case that \( f(n, m) \leq c g(n, m) \)

   (c) There exists a constant \( c > 0 \) such that

   \[
   \limsup_{n \to \infty} \limsup_{m \to \infty} \frac{f(n, m)}{g(n, m)} < c
   \]

   Note that this means that you first take the limit superior with respect to \( m \). The result will be a function of just \( n \). You then take the limit superior of this function with respect to \( n \). If you don’t know what limit superior means, you can just assume that the limit exists, in which case the limit and limit superior are the same.

   (d) There exists a constant \( c > 0 \) such that

   \[
   \limsup_{m \to \infty} \limsup_{n \to \infty} \frac{f(n, m)}{g(n, m)} < c
   \]

   Note that this means that you first take the limit superior with respect to \( n \). The result will be a function of just \( m \). You then take the limit superior of this function with respect to \( m \). If you don’t know what limit superior means, you can just assume that the limit exists, in which case the limit and limit superior are the same. Note that on the surface that this definition is different than the last on in that the order that you take the limits is switched.

   (e) There exists a constant \( c > 0 \) such that for all but finitely many pairs \((m, n)\) it is the case that \( f(n, m) \leq c g(n, m) \).
Due Monday January 9.

4. Problem 4-3 from the CLRS text except parts d and f. Apply the Master Theorem (Theorem 4.1) whenever applicable. Otherwise, draw the recursive call tree and sum up the costs level by level
   Due Wednesday January 11

5. Problem 4-3 parts d and f. Use induction; You will need 2 inductive proofs, one for the upper bound and one for the lower bound.
   Due Friday January 13.

6. Problem 4-6 parts d and e from the CLRS text.
   Due Friday January 13.

7. Problem 8.1-3 from the CLRS text. Give an adversarial strategy and prove that it is correct.
   Due Friday January 13.

8. Problem 8.1-4 from the CLRS text. Give an adversarial strategy and prove that it is correct. Explain the last sentence of the problem statement from the book, that is, explain why its not rigorous/correct to simply combine the lower bounds for the individual subsequences.
   Due Friday January 13.

9. Problem 8-6 from the CLRS text.
   (a) For part a and b, it is essentially asking you to consider the adversarial strategy that answers so as to maximize the number of the original ways of merging two sorted lists that are consistent with the answer. You will likely find Stirling’s approximation for n! useful.
   (b) For parts c and d, come up with a different adversarial strategy.
   (c) Explain why the bound that you get using the method proposed in parts a and b isn’t as good as the bound you get using an adversarial strategy. That is, in what way are you being too generous to the algorithm in parts a and b? In this generosity in the adversarial strategy or in its analysis?
   Due Monday January 23

10. Consider the problem of determining whether a collection of real numbers $x_1 \ldots x_n$ is nice. A collection of numbers is nice iff the difference between consecutive numbers in the sorted order is at most 1. So 1.2, 2.7, 1.8 is nice, but 1.2, 2.9, 1.8 is not nice since the difference between 1.8 and 2.9 is more than 1. We want to show that every comparison based algorithm to determine if a collection of n numbers is nice requires $\Omega(n \log n)$ comparisons of the form $x_i - x_j \leq c$, where c is some constant that the algorithm can specify. So if $c = 0$, this is a standard comparison.
11. Consider a setting where you have two computer networking routers $A$ and $B$. Each router has collected a list $L_A$ and $L_B$ of IP source addresses for the packets that have passed through the router that day. An IP address is $n$ bits, and thus there are $2^n$ possible IP addresses. Now the two routers want to communicate via a two-way channel to whether there was some source that sent a packet through one of the routers, but not the other. So more precisely, at the end of the protocol each router should commit to a bit specifying the answer to this question, and the bits for both routers should be correct. You can assume that each router can send a bit per unit time, and that a bit sent on the channel is guaranteed to arrive on the other end in one time unit. We want to consider protocols for accomplishing this goal.

(a) (warm-up) Consider the following protocol: $A$ sends to $B$ the list of all of the IP source addresses that it has seen; $B$ compares $A$’s list to its list, and then $B$ sends $A$ a 0 bit if the lists are identical and a 1 bit otherwise. Show that uses protocol about uses $n2^n + 1$ bits in the worst case.

(b) (warm-up) Give a protocol that uses $2^n + O(1)$ bits in the worst case. Another trivial warmup problem.

(c) (warm-up) Show that there is no protocol that can solve this problem without exchanging any bits.

Hint: Its obvious that this is true. That isn’t the point. The point is to understand what arguments you have to make to make this formally correct.

(d) Show that there is no protocol that can solve this problem that involves $A$ sending one bit to $B$. And no more bits are exchanged.

Hint: Again its obvious that this is true. Again that isn’t the point. Again the point is to understand what arguments you have to make to make this formally correct. Ask yourself how should the adversarial strategy should decide whether this first bit is a 0 or a 1?

(e) Show that there is no protocol that can solve this problem that involves $A$ sending one bit to $B$ and $B$ replying with one bit to $A$. And no more bits are exchanged.

Hint: Again its obvious that this is true. Again that isn’t the point. Again the point is to understand what arguments you have to make to make this formally correct.

(f) Prove that every protocol for this problem must send $2^n - O(1)$ bits for its worst case instance. Of course your argument should involve an adversarial argument.
(g) Assume that you have a computer networking router that sees a stream of $k$ IP packets, each with a source IP address consisting of $n$ bits. The router sees a packet, optionally records some information in memory, and then passes the packet on, then sees the next packet, optionally records some information in memory, and then passes that packet on, etc. The router’s goal is to always know an IP source address that it has seen most frequently to date. (The most obvious way to accomplish this is to keep a count for each IP source address seen to date.) Show that if $k \leq 2^n$ then every algorithm must use $\Omega(k)$ bits of memory.

Hint: This is an “easy” consequence of the previous subproblem, provided that you think about it the right way. Assume that you had a method that solved this problem using $o(k)$ bits of memory. Explain how to use this method to get an algorithm for the previous subproblem that uses less than $2^k$ bits of communication.

Due Wednesday January 25

12. Show that any comparison based algorithm for computing the median of $n$ numbers (you can assume for simplicity that $n$ is odd) requires $3n/2 - O(1)$ comparisons.

Hint: Some variation of the lower bound to compute the largest and smallest number will work, but there are a couple of additional complications.

Due Friday January 27

13. Problem 8-4 from the CLRS text. For part b give an adversarial strategy. For part c, use linearity of expectations in your algorithm analysis.

Due Friday January 27

14. Consider the problem of finding the largest $k$ numbers in sorted order from a list of $n$ numbers (see problem 9-1) in the text. Consider the following algorithm: you consider the numbers one by one, maintaining an auxiliary data structure of the largest $k$ numbers seen to date. We get various algorithms depending on what the auxiliary data structure is and how one searches and updates it. For each of the following variations give the worst-case time complexity as a function of $n$ and $k$. For each of the following variations give the average-case time complexity as a function of $n$ and $k$ under the assumption that each input permutation is equally likely.

Hint: Use linearity of expectations. These are all similar and easy if you look at them the right way.

(a) The auxiliary data structure is an ordered list and you use linear search starting from the end that contains the largest number

(b) The auxiliary data structure is an ordered list and you use linear search starting from the end that contains the smallest number

(c) The auxiliary data structure is a balanced binary search tree and you use standard log time search, insert and delete operations
(d) The auxiliary data structure is a balanced binary search tree and you use standard log time insert and delete operations, but you start your search from the smallest item in the tree.

Due Friday January 27

15. Consider the following problem. The input is \( n \) disjoint line segments contained in an \( L \) by \( L \) square \( S \) in the Euclidean plane. The goal is to partition \( S \) into convex polygons so that every polygon intersects at most one line segment. So it is ok for a line segment to be in multiple polygons, but each polygon can intersect at most one line segment.

Consider the following algorithm that starts with the polygon \( S \). Let \( \pi \) be a random permutation of the line segments. While there is a polygon \( P \) that contains more than one line segment, let \( \ell \) be the first line segment in the \( \pi \) order that intersects \( P \). Then cut \( P \) into two polygons using the linear extension of \( \ell \) (so you extend the line segment \( \ell \) into a line and then use that to cut \( P \) into two polygons). Show that the expected number of resulting polygons is \( O(n \log n) \).

Hint: Use linearity of expectations. First ask yourself how the number of polygons is related to the number of times that line segments get cut in the process. Consider to line segments \( u \) and \( v \). Let \( C_{u,v} \) be a 0/1 random variable that is 1 if the linear extension of \( u \) cuts \( v \). Let \( N(u,v) \) denote the number of line segments \( w \) where the linear extension of \( u \) hits \( w \) before hitting \( v \). In other words if you starting walking from \( u \), on \( u \)'s linear extension, towards \( v \), \( N(u,v) \) is how many line segments you cross before hitting \( v \). If you don’t hit \( v \), then \( N(u,v) = +\infty \). What is the relationship between the probability that \( C_{u,v} = 1 \) and \( N(u,v) \)?

Due Monday January 30

16. Assume you have a source of random bits. So in one time unit, this source will produce one random bit (that is 1 with probability \( 1/2 \) independent of other bits). Consider the problem of outputting a random permutation of the integers from 1 to \( n \). So each of the \( n! \) permutations should be produced with probability exactly \( 1/n! \).

(a) Give an algorithm to solve this problem and show that the expected time of the algorithm is \( O(n \log n) \). This includes both the time that your algorithm takes, plus 1 unit of time for each random bit used.

(b) Now assume that there is a limited source of at most \( n^2 \) random bits. Show that there is no algorithm that can solve the problem using expected time \( O(n^2) \).

Hint: Show the result for \( n=3 \). Why can’t you produce a random permutation of 1, 2, 3 using 9 bits? Then generalize to an arbitrary \( n \).

Further Hint: Often students will say that there two subproblems are contradictory. Maybe start by understanding why they are not contradictory.

Due Monday January 30.
17. 11-1 from the CLRS text.

Due Wednesday February 1

18. (a) Let $P$ be a problem. The worst case time complexity of $P$ is $O(n^2)$. The worst case time complexity of $P$ is $\omega(n \log n)$. Let $A$ be an algorithm that solves $P$. Which of the following statements are consistent with this information about the complexity of $P$.

i. $A$ has worst case time complexity $O(n^3)$.
ii. $A$ has worst case time complexity $O(n^{3/2})$.
iii. $A$ has worst case time complexity $O(n \log n)$.
iv. $A$ has worst case time complexity $\Theta(n^2)$.
v. $A$ has worst case time complexity $\Omega(n \log n)$.
vi. $A$ has worst case time complexity $\Omega(n)$.

(b) Let $P$ be a problem. The worst case time complexity of $P$ is $O(n^2)$. The worst case time complexity of $P$ is $\omega(n)$. Let $A$ be an algorithm that solves $P$. Which of the following statements are logically consistent with this information about the complexity of $P$.

i. $A$ has worst case time complexity $O(n^4)$.
ii. $A$ has worst case time complexity $\Theta(n^2)$.
iii. $A$ has worst case time complexity $O(n \log n)$.
iv. $A$ has worst case time complexity $O(n)$.
v. $A$ has worst case time complexity $\omega(n \log n)$.
vi. $A$ has worst case time complexity $\Omega(n \log n)$.

(c) Assume that problem $P$ has worst-case time complexity $o(n^3)$. Assume that problem $P$ has worst-case time complexity $\Omega(n^2)$. Assume that $A$ is an algorithm for problem $P$. For each of the following statements, state whether the statement is logically implied by the above information, and state whether the statement is logically consistent with the above information. So I am looking for 8 yes/no answers.

i. Algorithm $A$ has worst-cast time complexity $O(n^2)$.
ii. Algorithm $A$ has worst-cast time complexity $\Omega(n^3)$.
iii. Algorithm $A$ has worst-cast time complexity $\omega(n)$.
iv. Algorithm $A$ has worst-cast time complexity $o(n)$.

Due Friday February 3

19. 11-2 from the CLRS text.

Due Friday February 3

20. (a) Consider the problem where the input is a sorted array $A$ containing $n$ real numbers and a real number $x$, and the output is an integer $k$ such that if $x$ is in $A$
then it must lie between positions $k$ and $k + \sqrt{n}$. Assume that each element of $A$ and $x$ are independently and uniformly distributed in the interval $[0, 1]$. Show that the following algorithm solves this problem in $O(1)$ average case time.

```python
last = x*n
if A[last] < x then
    next = last + sqrt(n)
    while A[next] < x do
        last = next
        next = next + sqrt(n)
    return k = last
else if A[last] > x then
    next = last - sqrt(n)
    while A[next] > x do
        last = next
        next = next - sqrt(n)
    return k = next
```

HINT: Find the Bernoulli trials. Figure out how to think about the outcome of this algorithm in terms of the number of successes/failures in some Bernoulli trials. Use a Chernoff tail bound. See appendix C.5. You can use the result of exercise C.5-6 without proof.

(b) Explain how to use the above algorithm to obtain an algorithm with $O(\log \log n)$ average case running time for the searching problem (finding the location of the element in $A$ whose value is closest to $x$). Again assume that each element of $A$ and $x$ are independently and uniformly distributed in the interval $[0, 1]$.

Due Monday February 6

21. The purpose of this problem is to develop a version of Yao’s technique for Monte Carlo randomized algorithms, within the context of the red and blue jug problem from problem 8-4 in the CLRS text. Assume that if you sorted the jugs by volume, that each permutation is equally likely.

(a) Show that if a deterministic algorithm $A$ always stops in $o(n \log n)$ steps, then the probability that $A$ is correct for large $n$ is less than 1 percent.

(b) Show if there is a distribution of the input on which no deterministic algorithm with running time $A(n)$ is correct with probability $> 1$ percent, then there is no Monte Carlo algorithm with running time $A(n)$ that can be correct with probability $> 1$ percent.

Hint: Mimic the proof of Yao’s technique/lemma for the case of Las Vegas algorithms. Consider a two dimensional table/matrix $T$, where entry $T(A, I)$ is 1 if algorithm $A$ is correct on input $I$, and 0 otherwise.

(c) Conclude that any Monte Carlo algorithm for this jug problem must have time complexity $\Omega(n \log n)$. 


Due Wednesday February 8

22. Consider the following online problem. You given a sequence of bits $b_1,...b_n$ over time. Each bit is in an envelope. You first see the envelope for $b_1$, then the envelope for $b_2$, .... When you get the $i^{th}$ envelope, you can either look inside to see the bit, or destroy the envelope (in which case you will never know what the bit is). You know a priori that at least $n/2 + 1$ of the bits are 1. You goal is to find an envelope containing a 1 bit. You want to open as few envelopes as possible.

(a) Give a deterministic algorithm that will open at most $n/2 + O(1)$ envelopes.
HINT: This is completely straight-forward.

(b) Give an adversarial strategy to show that every deterministic algorithm must open at least $n/2 - O(1)$ envelopes.
HINT: This is completely straight-forward.

(c) Assume that each of the $n!$ permutations of the inputs is equally likely. Show that there is a deterministic algorithm where the expected number of envelopes that is opens is $O(1)$.
HINT: This is a straight-forward consequence of some facts that we learned about Bernoulli trials.

(d) Give a Monte Carlo algorithm that opens $O(\log n)$ envelopes and has probability of error $\leq 1/n$. Show that the probability of error is this small.
HINT: This is a straight-forward consequence of some facts that we learned about Bernoulli trials.

(e) Show using the version of Yao’s technique for Monte Carlo algorithms that you developed in the last homework assignment to show that every Monte Carlo algorithm must open $\Omega(\log n)$ envelopes if it is to be incorrect with probability $\leq 1/n$.
HINT: This is a straight-forward application of the Yao’s technique for Monte Carlo algorithms that you developed in the previous homework problem.

(f) Give a Las Vegas algorithm where the expected number of opened envelopes is $O(n^{1/2})$.
HINT: Take some random guesses for the first half of the envelopes, and then if you don’t find a 1 bit, give up and do the most obvious thing. See the discussion of the Birthday paradox in section 5.4.1. You may use facts from the analysis of the Birthday paradox in the CLRS text or from the wikipedia [http://en.wikipedia.org/wiki/Birthday_problem](http://en.wikipedia.org/wiki/Birthday_problem) without proof.

Due Friday February 10

23. Show that every Las Vegas algorithm for the previous envelope problem must open $\Omega(n^{1/2})$ envelopes in expectation.
HINT: Use Yao’s technique and the following probability distribution. With probability half, $\sqrt{n}$ uniformly distributed random bits in envelopes $[1, n/2]$ are set to 1(the other
bits in the first half of the envelopes are set to 0), bits in the envelopes \([n/2+1, n/2+\sqrt{n}]\) are all set to 0, and the remaining bits are 1. For \(k = 0, \ldots, \sqrt{n}\), with probability \(1/(2\sqrt{n})\), the bits in envelopes are distributed according to distribution \(D_k\). In \(D_k\), envelopes \([1, \ldots, n/2]\) contain a uniformly distributed random set of \(k\) 0’s and \(n/2-k\) 1’s. Then envelopes \([n/2 + 1, n/2 + \sqrt{n} + 1]\) contain a uniformly distributed set of \(\sqrt{n} - k\) 0’s and \(k + 1\) 1’s. The remaining bits are 0.

This is one of the hardest homework problems of the semester. Feel free to consider this to be extra credit if you like.

Due Monday February 13

24. Problem 16-5 part c. Use an exchange argument. The obvious exchange works, but the proof of optimality is a bit subtle. You can find some notes about what an exchange argument is here: [http://people.cs.pitt.edu/~kirk/cs1510/notes/greedynotes.pdf](http://people.cs.pitt.edu/~kirk/cs1510/notes/greedynotes.pdf)

Due Wednesday February 15


Hint: First figure out how to exhaustively generate all feasible solutions as the leaves of some tree. Then construct a pruning rule that you apply to each level of the tree to prune out some subtrees rooted at this level. The natural pruning rule is similar to the one used by Bellman-Ford.

Due Wednesday February 15

26. The input to this problem is two sequences \(T = t_1, \ldots, t_n\) and \(P = p_1, \ldots, p_k\) such that \(k \leq n\), and a positive integer cost \(c_i\) associated with each \(t_i\). The problem is to find a subsequence of of \(T\) that matches \(P\) with maximum aggregate cost. That is, find the sequence \(i_1 < \ldots < i_k\) such that for all \(j, 1 \leq j \leq k\), we have \(t_{i_j} = p_j\) and \(\sum_{j=1}^{k} c_{i_j}\) is maximized.

So for example, if \(n = 5\), \(T = XYXXY\), \(k = 2\), \(P = XY\), \(c_1 = c_2 = 2\), \(c_3 = 7\), \(c_4 = 1\) and \(c_5 = 1\), then the optimal solution is to pick the second \(X\) in \(T\) and the second \(Y\) in \(T\) for a cost of \(7 + 1 = 8\).

(a) Give a dynamic programming algorithm based on enumerating subsequences of \(T\) and using the pruning method.

(b) Give a dynamic programming algorithm based on enumerating subsequences of \(P\) and using the pruning method.

Due Friday February 17

27. There are three shortest path algorithms covered in chapter 24 (Bellman-Ford, Dijkstra, and the topological sort algorithm for directed acyclic graphs). For each of the following problems, pick the most appropriate of these three shortest path algorithms to apply to obtain an algorithm for the problem. This may or may not involve modifying the algorithm slightly. If you need to modify the algorithm, explain how. You may need to first briefly explain why the problem is indeed just a shortest path problem in
disguise; That is, state how one obtains the graph, and why the shortest path in this graph corresponds to a solution to the problem. Give the running time of the resulting algorithm.

(a) The problem described in 24-2
(b) The problem described in 24-3
(c) The problem described in 24-6
(d) The problem of finding the path where the minimum edge weight is maximized. You need such an algorithm to implement one of the Karp-Edmonds variations on Ford-Fulkerson.

Due Monday February 20

28. For each of the next 4 algorithms, state whether the algorithm is a polynomial time algorithm, whether the algorithm is a pseudo-polynomial time algorithm, and whether the algorithm is a strongly polynomial time algorithm. Justify your answers.

(a) Read($x_1, \ldots, x_n$)
   \begin{verbatim}
   y=2
   For i := 1 to n do
       For i := 1 to n do y:=y + x_j * x_i
   \end{verbatim}
(b) Read($x_1, \ldots, x_n$)
   \begin{verbatim}
   y=2
   For i := 1 to n do
       For i := 1 to $x_i$ do y:=y + x_j * x_i
   \end{verbatim}
(c) Read($x_1, \ldots, x_n$)
   \begin{verbatim}
   y=2
   For i := 1 to n do
       For i := 1 to log $x_i$ do y:=y + x_j * x_i
   \end{verbatim}
(d) Read($x_1, \ldots, x_n$)
   \begin{verbatim}
   y=2
   For i := 1 to n do
       For i := 1 to n do y:=y * y
   \end{verbatim}
   Hint: This one is a bit tricky. If the answer is not clear, explain why.

Due Monday February 20

29. Show how each of problems described problems 26-1, 26-2 and 26-3 from the CLRS text can be efficiently reduced to network flow. Give the running time of the resulting algorithms for each problem assuming that you can solve network flow in time $T(V, E, F)$, where $T$ is some function of the number of vertices $V$ in the network, the number of edges $E$ in the network, and the max flow $F$ in the network.

Due Wednesday February 22
30. Problem 26-5 from the CLRS text.
   Due Wednesday February 22

31. Consider the problem of constructing a maximum cardinality bipartite matching. The input is a bipartite graph, where one bipartition are the girls, and one bipartition is the boys. There is an edge between a boy and a girl if they are willing to dance together. The problem is to match the boys and girls for one dance so that as many couples are dancing as possible. See section 26.3 in the book if you want more details.

   (a) Construct an integer linear program for this problem. So you want to explain how to compute maximum cardinality matchings by finding an optimal integer solution to a particular linear program.

   (b) Consider the relaxed linear program where the integrality requirements are dropped. Explain how to find an integer optimal solution from any rational optimal solution. Hint: Find cycles of edges with associated variables that are not integer.

   Due Friday February 24

32. Consider the minimum spanning tree problem defined in Chapter 23 of the text.

   (a) Give an integer linear programming formulation using the following intuition, and prove that your formulation is correct: There is an indicator 0/1 random variable for each edge. You must choose at least $n - 1$ edges ($n$ is the number of vertices in the graph). For each subset $S$ of $k$ vertices, you can choose at most $k - 1$ edges connecting vertices in $S$. Explain why the size of this linear program can be exponential in the size of the graph.

   (b) Give an integer linear programming formulation using the following intuition, and prove that your formulation is correct: There is an indicator 0/1 random variable for each edge. You must choose at exactly $n - 1$ edges. For each subset $S$ of vertices ($S$ not the empty set and not all the vertices), you must choose at least one edge with one endpoint in $S$ and one endpoint not in $S$. Explain why the size of this linear program can be exponential in the size of the graph. HINT: Theorem 23.1 in the text may be useful.

   (c) Give a polynomial sized integer linear programming formulation using the following intuition, and prove that your formulation is correct: Call an arbitrary vertex the root $r$. Think of a spanning tree as routing flow away from $r$ to the rest of the tree (but now you do not have flow conservation at the vertices). Explain why the size of this linear program is polynomially bounded in the size of the graph.

   (d) Consider a relaxation of the integer linear program in the last subproblem in that now the flows on the edges may be rational (and not necessarily integer). Show how to express an optimal rational solution to the linear program as an affine combination of optimal (rooted) spanning trees. Conclude that one can compute in polynomial time an optimal spanning given the optimal rational solution to this linear program.
HINT: The coefficient for the first tree will be the least flow on any edge. And then repeat this idea.

Due Friday February 24

33. Consider the following problem. The input consists of a directed graph $G$ with a specified source vertex $s$, and a positive integer profit for each vertex of $G$. The objective is to find a subset $H$ of the vertices of maximum total profit subject to the constraint that there is a collection of vertex disjoint paths from $s$ to the vertices in $H$. That is, if $H = \{v_1, \ldots, v_k\}$ there must be paths $P_1, \ldots, P_k$ such that each path $P_i$ starts at $s$ and ends at $v_i$, and no pair of paths share a vertex other than $s$.

(a) Write an integer linear programming formulation for this problem
   Hint: As is usually the case, the key is to figure out what the variables will be. Probably the most natural formulation has two types of variables.

(b) Consider the relaxed linear program where the integrality requirements are dropped. Explain how to find an integer optimal solution from any rational optimal solution.
   Hint: Morally this rounding is the same as for matching, but the implementation is a bit more complicated.

(c) Give a strongly polynomial time algorithm for this problem (note that no one knows of a strongly polynomial time algorithm for linear programming). You must prove your algorithm is both correct, and runs in strongly polynomial time.
   Hint: What is Gordon Gekko’s most famous quote? You can Google it if you don’t know.

Due Monday February 27

34. Consider a two person game specified by an $m$ by $n$ payoff matrix $P$. The two players can be thought of as a row player and a column player. The number of possible moves for the row player is $m$ and the number of possible moves for the column player is $n$. Each player picks one of its moves, and then money is exchanged. If the row player makes move $r$, and the column player makes move $c$, then the row player pays the column player $P_{r,c}$ dollars. Note that $P_{r,c}$ could be negative, in which case really the column player is paying money to the row player. We assume that the game is played sequentially, so that one player specifies his move, the other players sees that move, and then specifies a response move (we can assume that this player makes the best possible response). Obviously each player wants to be payed as much money as possible, and if this is not possible, to pay as little as possible.

(a) Give a simple/efficient algorithm that will compute the best response for the column player give a specific move by the row player.

(b) Give a simple/efficient algorithm that will compute the best first move by the row player given that the column player will give its best response.
(c) Either give an example of a payoff matrix where it is strictly better for each player to go second, or argue that there is no such payoff matrix.
HINT: Roshambo

Now we change the problem so that each player specifies a probability distribution over his moves, and then the row player pays the column player $E[P_{r,c}]$, where the expectation is taken over the two probability distributions in the natural way.

(d) Give a simple/efficient algorithm that will efficiently compute the best response (which is probability distribution over column moves) for the column player given a probability distribution specified by the row player.

(e) Give a linear program that will compute the best first move (probability distribution over row moves) for the row player given that the column player makes the best response.

(f) Show the linear program you would get for the following payoff matrix:

\[
\begin{array}{ccc}
2 & 3 & 5 \\
11 & 7 & 4 \\
6 & 9 & 1 \\
\end{array}
\]

(g) Give a linear program to compute the best first move (probability distribution over column moves) for the column player given that the row player makes the best response.

(h) Show the linear program you would get for the following payoff matrix:

\[
\begin{array}{ccc}
2 & 3 & 5 \\
11 & 7 & 4 \\
6 & 9 & 1 \\
\end{array}
\]

(i) Either give an example of a payoff matrix where it is strictly better for each player to go second, or argue that there is no such payoff matrix.
Hint: Strong linear programming duality.

Due Wednesday March 1

35. Consider the problem of constructing maximum cardinality bipartite matching (See section 26.3 in the CLR text).

(a) Construct an integer linear program for this problem

(b) Construct the dual linear program.

(c) Give a natural English interpretation of the dual problem (e.g. similar to how we interpreted the dual of diet problem as the pill problem)
(d) Explain how to give a simple proof that a graph doesn’t have a matching of a particular size. You should be able to come up with a method that would convince someone who knows nothing about linear programming.

Due Wednesday March 1

36. Consider the problem of scheduling a collection of processes on one processor. Each process $J_i$ has a work $x_i$, a release time $r_i$, and a deadline $d_i$. All these values are positive integers. A job can not be run before its release time or after its deadline. The goal is to find the slowest possible speed that will allow you to finish do all the work from each job. If a job is processed at speed $s$ for $t$ units of time, this will complete $s \cdot t$ units of work from that job. The processor can process at most one job at each moment of time. But a processor can switch between processes arbitrarily. For example, the processor can run $J_1$ for a while, then switch to $J_2$, then back to $J_1$, then to $J_3$, etc. The times of these switches need not be integer.

(a) Express this problem as a linear program.
(b) Construct the dual program.
(c) Give a natural English interpretation of the dual problem (e.g. similar to how we interpreted the dual of the max flow problem as the min cut problem).
(d) Explain how to give a simple proof that the input is infeasible for a particular speed. You should be able to come up with a method that would convince someone who knows nothing about linear programming.

Hint: Find the relationship between this problem and the previous problem. That is, figure out what this problem has to do with bipartite matching. You are welcome to use facts you developed in solving the previous problem.

Due Friday March 3

37. Assume that you have a park (mathematically a 2 dimensional plane) containing $k$ lights and $n$ statues. In particular, you know for each light $L$ and for each statue $S$, whether light $L$ will illuminate statue $S$ if light $L$ is lit. This is a binary value, there is no possibility of partial illumination. Further you are told for each light $L$, the cost $C_L$ for turning $L$ on. The goal is to light all the statues while spending as little money as possible.

(a) Construct an integer linear program for this problem where there are binary indicator variables for each light signifying whether the light is lit or not.
(b) Consider the relaxed linear program where the variables are allowed to be any rational between 0 and 1. Give an English explanation of the problem that this models.

HINT: Imagine the lights have a dimmer control.
(c) Show that the relaxed linear program where the variables are allowed to be any rational between 0 and 1 can have a strictly smaller objective than the optimal objective for the integer linear program for some instances.
(d) Construct the dual program for the relaxed linear program.

(e) Give a natural English interpretation of the dual problem (the problem modeled by the dual linear program)

(f) Explain how to give a simple proof that a certain cost is required for the problem modeled by the relaxed linear program (the one with dimmer controls) using this natural interpretation of the dual.

Due Monday March 13

38. Prove that each of the problems defined in 34.5-1, 34.5-2, 34.5-3, 34.5-5, 34.5-6, 34.5-7, and 34.5-8 are NP-hard using a reduction using a reduction from an NP-complete problem of your choice that is defined earlier in Chapter 34. So for each problem, you need to give one polynomial time reduction. The difficulty of finding the reductions ranges from trivial to reasonably straight-forward.

Due Wednesday March 15

39. Show that the 3-COLOR problem is NP-hard by reduction from the 3-CNF-SAT problem. 3-COLOR is defined in problem 34-3 in the text, which also contains copious hints.

Due Friday March 17

40. We consider a generalization of the Fox, goose and bag of beans puzzle [15].

The input is a graph G an integer \( k \). The vertices of G are objects that the farmer has to transport over the river, there are an edge between two objects if they can not be left alone (without the farmer’s supervision) on the same size of the river. The goal is to determine if a boat of size \( k \) is sufficient to safely transport the objects across the river. The size of the boat is the number of objects that the farmer can haul in the boat.

Show that this problem is NP-hard using a reduction from one of the problems depicted in figure 34.13 in the CLRS text. So I am letting you pick the problem to reduce from here. You should take some time to reflect which problem would be easiest to reduce from.

Due Monday March 20

41. Problem 35-5 parts a, b and d from the CLRS text.

Due Monday March 20

42. 35-7 from the CLRS text

Due Wednesday March 22

43. Problem 35.2-3 from the CLRS text.

Due Friday March 24
44. Problem 35.2-4 from the CLRS text.
   Due Friday March 24

45. Prove that if there is a polynomial time approximation algorithm for the maximum clique problem that has approximation ratio 1000 then there is a polynomial time approximation algorithm with approximation ratio 1.000000001. This is actually a slightly easier problem than problem 35-2 part b in the CLRS text, which I suggest that you look at for inspiration. Note that in some sense this can be viewed as a gap reduction.
   Due Monday March 27

46. Consider the following problem. The input is a graph $G = (V,E)$. Feasible solutions are subsets $S$ of the vertices $V$. The objective is to maximize the number of edges with one endpoint in $S$ and one endpoint in $V - S$.

   (a) Give a simple polynomial-time randomized algorithm for this problem and show that it is 2 approximate.
   Hint: Flip a coin for vertex and consider analysis for MAX2SAT from class.

   (b) Develop a deterministic polynomial-time 2-approximation algorithm for this problem using the method of conditional expectations, which considers the vertices one by one, but instead of flipping a coin for each vertex $v$, puts $v$ in the bipartition that would maximize the expected number of edges in the cut if coin flips were used for the remaining vertices.

   (c) Give a simple greedy algorithm that ends up implementing this policy.

   (d) Prove that this greedy algorithm has approximation ratio at most 2.
   Due Monday March 27


   (a) Given an integer linear programming formulation of MAX-SAT.
   Hint: There should be two types of variables in the linear program: One linear programming variable $x_v$ for each variable $v$ in the MAX-SAT instance, and one linear programming variable $y_c$ for each clause $c$ in the MAX-SAT instance.

   (b) Let $x_v^*$ be an optimal solution to the relaxed (rational) linear program. Show that setting the variable $v$ to 1 with probability $x_v^*$, and to 0 otherwise (independent of the setting of the other variables) yields a $(1 - \frac{1}{e})$-approximation.
   Hint: Use the fact that the geometric mean is at most the arithmetic mean [https://en.wikipedia.org/wiki/Inequality_of_arithmetic_and_geometric_means](https://en.wikipedia.org/wiki/Inequality_of_arithmetic_and_geometric_means).
   Also use the fact that if $f(x)$ is a concave function on an interval $[a,b]$, you can lower bound $f(x)$ in this interval the line that passes through $(a,f(a))$ and $(b,f(b))$.  ```
Due Wednesday March 29

48. Problem 17.3-6 from the CLRS text. You must use a potential function analysis to prove $O(1)$ amortized time.
   Hint: The potential function for dynamic tables will be useful.

Due Friday March 31

49. Problem 17-3 from the CLRS text.

Due Friday March 31

50. Assume that you have a collection of $n$ boxes arriving online over time that must be loaded onto $m$ trucks. When a box arrives, the online algorithm learns the weight of the box, and a list of trucks that that box can be loaded on. So not every box is allowed to be loaded on every truck. At the time that a box arrives, the online algorithm must pick a truck to load the box on. The objective is to minimize the weight of the most heavily loaded truck. Give an adversarial argument to show no deterministic online algorithm can achieve approximation ratio $O(1)$.
   Hint: In your adversarial strategy, later arriving boxes should be made only assignable to trucks that the online algorithm assigned boxes to earlier.

Due Monday April 3

51. Consider the paging problem. Consider the following randomized online algorithm.

   **Algorithm Description:** Each page $P$ has an associated bit denoting whether the page is FRESH or STALE. If requested page $P$ in fast memory, then $P$’s associated bit is set to FRESH. If the requested page $P$ is not in fast memory, then a STALE page is selected uniformly at random from the STALE pages in fast memory and ejected, and $P$’s associated bit is set to FRESH. If the request page $P$ is not in fast memory, and all pages in fast memory are FRESH, then make all pages in fast memory STALE, select a STALE page uniformly at random from the STALE pages in fast memory to evict, and $P$ associated bit is set to FRESH.

   Show that this algorithm is $O(\log k)$ competitive/approximate using the following strategy (recall $k$ is the size of the fast memory). Partition the input sequence into consecutive subsequences/phases where there are exactly $k$ distinct pages requested in each subsequence/phase. The phase breaks are when all pages in fast memory are made STALE. Let $m_i$ be the number of pages requested in phase $i$ that were not requested in phase $i - 1$.

   (a) Show that the optimal number of page faults is $\Omega(\sum_i m_i)$.

   (b) Show that the expected number of page faults for the randomized algorithm on the page requests in phase $i$ is $O(m_i \log k)$.

Due Wednesday April 5
52. Consider an online or approximation problem where there are only finitely many possible algorithms and finitely many possible inputs. We generalize Yao’s technique to approximation ratios. The correct answer is “yes” to three of the following four questions, and the correct answer is “no” for the remaining question. Identify the three questions where the answer is yes, and give a proof that the answer is yes.

Hint: First assume that there are two possible deterministic algorithms and two possible inputs. The general proof is more or less the same, but it may be easier to think about this special case.

(a) Assume that the problem is a minimization problem
   i. Assume that you have an input distribution $I$, such that for all deterministic algorithms $A$ it is the case that $E[A(I)]/E[Opt(I)] \geq c$. Can you logically conclude that the expected competitive ratio for every randomized algorithm is at least $c$?
   ii. Assume that you have an input distribution $I$, such that for all deterministic algorithms $A$ it is the case that $E[A(I)/Opt(I)] \geq c$. Can you logically conclude that the expected competitive ratio for every randomized algorithm is at least $c$?

(b) Assume that the problem is a maximization problem
   i. Assume that you have an input distribution $I$, such that for all deterministic algorithms $A$ it is the case that $E[Opt(I)]/E[A(I)] \geq c$. Can you logically conclude that the expected competitive ratio for every randomized algorithm is at least $c$?
   ii. Assume that you have an input distribution $I$, such that for all deterministic algorithms $A$ it is the case that $E[Opt(I)/A(I)] \geq c$. Can you logically conclude that the expected competitive ratio for every randomized algorithm is at least $c$?

(c) For extra credit, prove that the correct answer is no for the remaining question.
   There is a simple example, but that does not mean it is easy to find

Due Friday April 7

53. Show that the expected competitive ratio for every randomized paging algorithms is $\Omega(\log k)$.

Hint: Use your results from the previous problem. Assume that the number of pages is one more than the size of fast memory.

Due Monday April 10

54. Consider the following online problem. There are two taxis on a line that initially start at the origin. At positive integer time $t$, a request point $h_t$ on the line arrives. In response, each taxi can move to a different location on the line, or stay put at its current point. The path traveled by at least one of the two taxis much cross $h_t$. The objective is to minimize the total movement of the taxis.
(a) As a warmup show that if there is a $c$-competitive algorithm $A$ for this problem, then there is a $c$-competitive algorithm $B$ that only moves one taxi in response to each request, and that one taxi moves directly from its position to the request.

(b) Give an adversarial strategy to show that the competitive ratio of every deterministic algorithm is at least 2.

Hint: Come up with a request sequence that makes it hard to decide if one of the taxis should move.

(c) Consider the following algorithm $A$. If both taxis are to the left of $h_t$, then a rightmost taxi moves to $h_t$. If both taxis are to the right of $h_t$, then a leftmost taxi moves to $h_t$. If $h_t$ is between the two taxis, then both taxis move toward $h_t$ at the same rate until one of the taxis reaches $h_t$, at which point both taxis stop moving. Show that this algorithm is 2-competitive using the following potential function: $\Phi = \text{twice the distance between the leftmost taxi for } A \text{ and the leftmost taxi for optimal} + \text{twice the distance between the rightmost taxi for } A \text{ and the rightmost taxi for optimal} + \text{the distance between the leftmost and the rightmost taxis for } A$. So you need to show that for each request, the cost to $A$ + the change in the potential $\Phi$ is at most 2 times the cost to optimal.

Due Wednesday April 12