Quiz 1

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 120 minutes to earn 120 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- This quiz is closed book. You may use **one** $8\frac{1}{2}'' \times 11''$ or A4 crib sheet (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required.
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	2	2		
2	3	20		
3	9	18		
4	2	20		
5	2	20		
6	3	20		
7	2	20		
Total		120		

Name: _____

Wed/Fri Recitation:	Ying	Kevin	Sarah	Yafim	Victor
	10, 11 AM	**11 AM**	**12, 1 PM**	**12 PM**	**2, 3 PM**

Problem 1. [2 points] Write your name on top of each page.

Problem 2. Asymptotics & Recurrences [20 points] (3 parts)

(a) [10 points] Rank the following functions by *increasing* order of growth. That is, find any arrangement $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$ of the functions satisfying $g_1 = O(g_2)$, $g_2 = O(g_3)$, $g_3 = O(g_4)$, $g_4 = O(g_5)$, $g_5 = O(g_6)$, $g_6 = O(g_7)$, $g_7 = O(g_8)$.

$$f_1(n) = n^{\pi} \qquad f_2(n) = \pi^n \qquad f_3(n) = \binom{n}{5} \qquad f_4(n) = \sqrt{2^{\sqrt{n}}}$$

$$f_5(n) = \binom{n}{n-4} \qquad f_6(n) = 2^{\log^4 n} \qquad f_7(n) = n^{5(\log n)^2} \qquad f_8(n) = n^4\binom{n}{4}$$

(b) [5 points] Find a solution to the recurrence $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n)$.

(c) [5 points] Find an asymptotic solution of the following recurrence. Express your answer using Θ-notation, and give a brief justification.

$$T(n) = \log n + T\left(\sqrt{n}\right)$$

Problem 3. True/False [18 points] (9 parts)

Circle (T)rue or (F)alse. You don't need to justify your choice.

(a) T F [2 points] Binary insertion sorting (insertion sort that uses binary search to find each insertion point) requires $O(n \log n)$ total operations.

(b) T F [2 points] In the merge-sort execution tree, roughly the same amount of work is done at each level of the tree.

(c) T F [2 points] In a BST, we can find the next smallest element to a given element in $O(1)$ time.

(d) T F [2 points] In an AVL tree, during the insert operation there are at most two rotations needed.

(e) T F [2 points] Counting sort is a stable, in-place sorting algorithm.

(f) T F [2 points] In a min-heap, the next largest element of any element can be found in $O(\log n)$ time.

(g) T F [2 points] The multiplication method satisfies the simple uniform hashing assumption.

(h) T F [2 points] Double hashing satisfies the uniform hashing assumption.

(i) T F [2 points] Python generators can be used to iterate over potentially infinite countable sets with $O(1)$ memory.

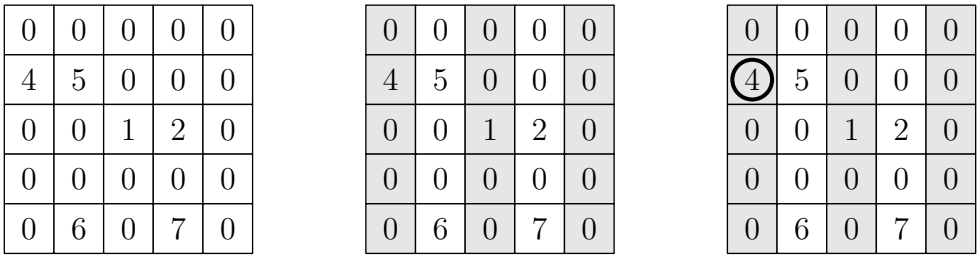
Problem 4. Peak Finding (again!) [20 points] (2 parts)

When Alyssa P. Hacker did the first 6.006 problem set this semester, she didn't particularly like any of the 2-D peak-finding algorithms. A peak is defined as any location that has a value at least as large as all four of its neighbors.

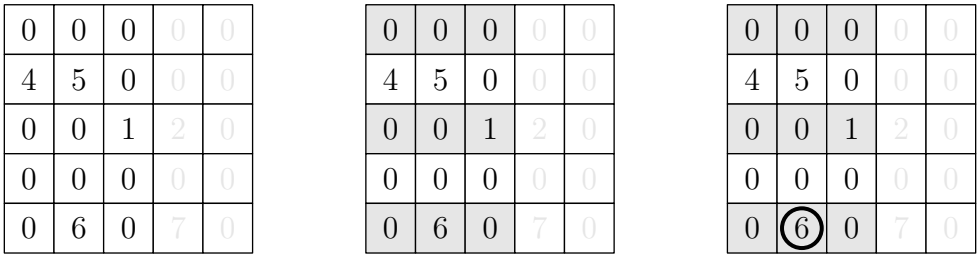
Alyssa is excited about the following algorithm:

1. Examine all of the values in the first, middle, and last columns of the matrix to find the maximum location ℓ.
2. If ℓ is a peak within the current subproblem, return it. Otherwise, it must have a neighbor p that is strictly greater.
3. If p lies to the left of the central column, restrict the problem matrix to the left half of the matrix, including the first and middle columns. If p lies to the right of the central column, restrict the problem matrix to the right half of the matrix, including the middle and last columns.
4. Repeat steps 1 through 3 looking at the first, middle, and last **rows**.
5. Repeat steps 1 through 4 until a peak is found.

Consider the 5×5 example depicted below. On this example, the algorithm initially examines the first, third, and fifth columns, and finds the maximum in all three. In this case, the maximum is the number 4. The number 4 is not a peak, due to its neighbor 5.



The number 5 is to the left of the middle column, so we restrict our view to just the left half of the matrix. (Note that we include both the first and middle columns.) Because we examined columns in the previous step, we now examine the first, middle, and last rows of the submatrix. The largest value still visible in those rows is 6, which is a peak within the subproblem. Hence, the algorithm will find the peak 6.



(a) [5 points] What is the worst-case runtime of Alyssa's algorithm on an $m \times n$ matrix (m rows, n columns), in big-Θ notation? Give a brief justification for your answer.

(b) [15 points] Does Alyssa's algorithm return a peak in all cases? If so, give a short proof of correctness. Otherwise, provide a counterexample for the algorithm.

Problem 5. Who Let The Zombies Out? [20 points] (2 parts)

In an attempt to take over Earth, evil aliens have contaminated certain water supplies with a virus that transforms humans into flesh-craving zombies. To track down the aliens, the Center for Disease Control needs to determine the epicenters of the outbreak—which water supplies have been contaminated. There are n potentially infected cities $C = \{c_1, c_2, \ldots, c_n\}$, but the FBI is certain that only k cities have contaminated water supplies.

Unfortunately, the only known test to determine the contamination of a city's water supply is to serve some of that water to a human and see whether they turn ravenous. Several brave volunteers have offered to undergo such an experiment, but they are only willing to try their luck once. Each volunteer is willing to drink a single glass of water that mixes together samples of water from any subset $C' \subseteq C$ of the n cities, which reveals whether at least one city in C' had contaminated water.

Your goal is to use the fewest possible experiments (volunteers) in order to determine, for each city c_i, whether its water was contaminated, under the assumption that exactly k cities have contaminated water. You can design each experiment based on the results of all preceding experiments.

(a) [10 points] You observe that, as in the comparison model, any algorithm can be viewed as a decision tree where a node corresponds to an experiment with two outcomes (contaminated or not) and thus two children. Prove a lower bound of $\Omega(k \lg \frac{n}{k})$ on the number of experiments that must be done to save the world. Assume that $\lg x! \sim x \lg x$ and that $\lg(n-k) \sim \lg n$ (which is reasonable when $k < 0.99n$).

(b) [10 points] Save the world by designing an algorithm to determine which k of the n cities have contaminated water supplies using $O(k \lg n)$ experiments. Describe and analyze your algorithm.

Problem 6. Shopping Madness [20 points] (3 parts)

Ben Bitdiddle was peer-pressured into signing up for the tryouts in a shopping reality TV show, and he needs your help to make it past the first round. In order to qualify, Ben must browse a store's inventory, which has N items with different positive prices $P[1], P[2], \ldots, P[N]$, and the challenge is to spend exactly S dollars on exactly K items, where K is a small even integer. Ben can buy the same item multiple times. For example, 3 brooms and 2 wizard hats add up to 5 items.

In your solutions below, you may use a subroutine MULTISETS$(k, \mathbb{T})$ which iterates over all the k-element multisets (like subsets, except the same elements can show up multiple times) of a set $\mathbb{T}$, in time $O(k \cdot |\mathbb{T}|^k)$, using $O(k)$ total space. Note that if your code holds onto the results of MULTISETS, it may end up using more than $O(k)$ space.

(a) [5 points] Write pseudo-code for a data structure that supports the following two operations.

INIT(N, K, P) — preprocesses the $P[1 \ldots N]$ array of prices, in $O(K \cdot N^K)$ expected time, using $O(K \cdot N^K)$ space, to be able to answer the query below.

BAG(S) — in $O(1)$ expected time, determines whether K of the items have prices summing to S, and if so, returns K indices $b_1, b_2, \ldots, b_K$ such that $S = \sum_{i=1}^{K} P[b_i]$.

(b) [10 points] Write pseudo-code for a function PWN-CONTEST(N, S, K, P) that determines whether K of the items have prices summing to S, and if so, returns K indices $b_1, b_2, \ldots, b_K$ such that $S = \sum_{i=1}^{K} P[b_i]$. Unlike part (a), PWN-CONTEST should run in $O(K \cdot N^{K/2})$ and use $O(K \cdot N^{K/2})$ space.

(c) [5 points] Analyze the running time of your pseudo-code for the previous part.

Problem 7. When I Was Your Age... [20 points] (2 parts)

In order to design a new joke for your standup comedy routine, you've collected n distinct measurements into an array $A[1 \ldots n]$, where $A[i]$ represents a measurement at time i. Your goal is to find the longest timespan $i \ldots j$, i.e., maximize $j - i$, such that $A[i] < A[j]$. [1] Note that the values in between $A[i]$ and $A[j]$ do not matter. As an example, consider the following array $A[1 \ldots 7]$:

$$A[1] = 14 \quad A[2] = 6 \quad A[3] = 8 \quad A[4] = 1 \quad A[5] = 12 \quad A[6] = 7 \quad A[7] = 5$$

Your algorithm should return a span of 4 since $A[2] = 6$ and $A[6] = 7$. The next biggest span is $A[4] = 1$ to $A[7] = 5$.

(a) [5 points] Give an $O(n)$-time algorithm to compute the minimums of the prefix $A[1 \ldots k]$ for each k, and store in $MA[k]$: $MA[k] = \min_{i=1}^{k} A[i]$.

(b) [15 points] Using the $MA[i]$ computed above, give an $O(n \log n)$-time algorithm to maximize $j - i$ subject to $A[i] < A[j]$.
Hint: The MA is a sorted array.

[1] The joke could be along these lines: "You thought time j was bad with $A[j]$? Back in time i, we only had $A[i]$!"

6.006 Introduction to Algorithms
Fall 2011

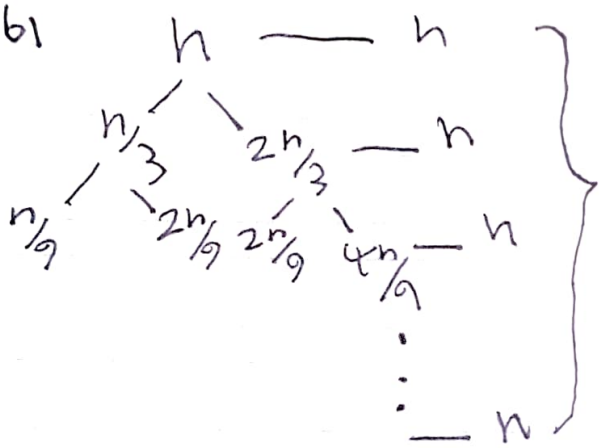
Subject: Quiz 1

Year: Month: Date:

Problem 1 Benyamin Lazayeri

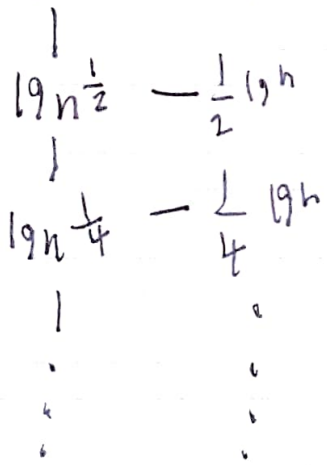
Problem 2

(a) $f_1, f_5, f_3, f_8, f_7, f_6, f_4, f_2$ ✓

(b) 

$\log_3 n$

$\Theta(n \log_3 n)$ ✓

(c) 

$\Theta(\lg n)$ ✓

Subject:

Year: Month: Date:

Problem 3

(a) F ✓

(b) T ✓

(c) F ✓

(d) F ✗ -2

(e) F ✓

(f) F ✓

(g) T ✗ -2

(h) T ✗ -2

(i) T ✓

Subject: Year: Month: Date:

Problem 4 -20

(a) $T(m/n) = 3n + T(m/2, n) = \Theta(n \lg m)$ X

(b) The algorithm is correct. X

We provide a 2 part Proof.

1. The Algorithm will terminate.

each sub problem has size

$\lfloor \frac{m}{2} \rfloor \times n$ or $m - \lfloor \frac{m}{2} \rfloor \times n$

So either returns or empty.

for empty, $m = 1 \rightarrow$ max will be Peak ✓

2. The returned value will be Peak. assume for ※ it being Passed up stops being peak → b is adjacent to middle column s.t. a<b and c is max on there → b<c and d is c neighbor d>c but a is max a>d ※ □

Subject:

Year: Month: Date:

Problem 5

(a) $\binom{n}{k}$ leaves. ✓

$$\lg\binom{n}{k} = n\lg n - n\lg n + k\lg n - k\lg k = k\lg \frac{n}{k}$$

(b) divide into $n/2$, test, recurse into positive worst case $k\lg n/k$. ✓

Problem 6 dict = {} ✓

(a)
```
for i in multisets(k, P):
    dict[sum(i)] = i
return dict[S]
```

dict = {} ✓

(b)
```
for i in multisets(k/2, P):
    dict[sum(i)] = i
for i in multisets(k/2, P):
    if S-sum(i) in dict:
        return i + dict[S-sum(i)]
```

c) $O(k \cdot n^{k/2})$

Subject: Year: Month: Date:

Problem 7

(a) if A[i] < min:
    min = A[i] ✓

(b) ~~Binary search for MA [i] < A[j]~~ ✗

-15

~~9~~ 7/12 = 65%

Quiz 2

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 120 minutes to earn 120 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- This quiz is closed book. You may use **two** $8\frac{1}{2}'' \times 11''$ or A4 crib sheets (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required.
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	1	1		
2	14	28		
3	5	26		
4	5	25		
5	1	15		
6	3	25		
Total		120		

Name: ____

Wed/Fri Recitation:	Ying **10, 11 AM**	Kevin **11 AM**	Sarah **12, 1 PM**	Yafim **12 PM**	Victor **2, 3 PM**

Problem 1. [1 points] Write your name on top of each page.

Problem 2. True/False [28 points] (14 parts)

Circle (T)rue or (F)alse. You don't need to justify your choice.

(a) T F [2 points] Computing $\lfloor \sqrt{a} \rfloor$ for an n-bit positive integer a can be done in $O(\lg n)$ iterations of Newton's method.

(b) T F [2 points] Suppose we want to solve a polynomial equation $f(x) = 0$. While our choice of initial approximation x_0 will affect how quickly Newton's method converges, it will always converge eventually.

(c) T F [2 points] Karatsuba's integer multiplication algorithm always runs faster than the grade-school integer multiplication algorithm.

(d) T F [2 points] If we convert an n-digit base-256 number into base 2, the resulting number of digits is $\Theta(n^2)$.

(e) T F [2 points] In a weighted undirected graph $G = (V, E, w)$, breadth-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(f) T F [2 points] In a weighted undirected **tree** $G = (V, E, w)$, breadth-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(g) T F [2 points] In a weighted undirected **tree** $G = (V, E, w)$, **depth**-first search from a vertex s finds single-source shortest paths from s (via parent pointers) in $O(V + E)$ time.

(h) T F [2 points] If a graph represents tasks and their interdependencies (i.e., an edge (u, v) indicates that u must happen before v happens), then the breadth-first search order of vertices is a valid order in which to tackle the tasks.

(i) T F [2 points] Dijkstra's shortest-path algorithm may relax an edge more than once in a graph with a cycle.

(j) T F [2 points] Given a weighted directed graph $G = (V, E, w)$ and a source $s \in V$, if G has a negative-weight cycle somewhere, then the Bellman-Ford algorithm will necessarily compute an incorrect result for some $\delta(s, v)$.

(k) T F [2 points] In a weighted directed graph $G = (V, E, w)$ containing no zero- or positive-weight cycles, Bellman-Ford can find a *longest* (maximum-weight) path from vertex s to vertex t.

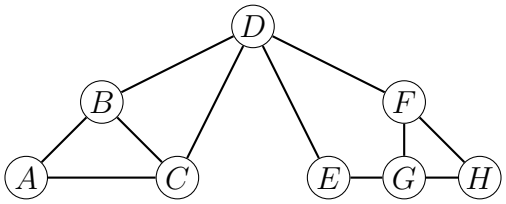
(l) T F [2 points] In a weighted directed graph $G = (V, E, w)$ containing a negative-weight cycle, running the Bellman-Ford algorithm from s will find a shortest acyclic path from s to a given destination vertex t.

(m) T F [2 points] The bidirectional Dijkstra algorithm runs asymptotically faster than the Dijkstra algorithm.

(n) T F [2 points] Given a weighted directed graph $G = (V, E, w)$ and a shortest path p from s to t, if we doubled the weight of every edge to produce $G' = (V, E, w')$, then p is also a shortest path in G'.

Problem 3. MazeCraft [26 points] (5 parts)

You are playing SnowStorm's new video game, *MazeCraft*. Realizing that you can convert a maze into a graph with vertices representing cells and edges representing passages, you want to use your newly learned graph-search algorithms to navigate the maze. Consider the following converted graph.



For the following questions, assume that the graph is represented using adjacency lists, and that **all adjacency lists are sorted**, i.e., the vertices in an adjacency list are always sorted alphabetically.

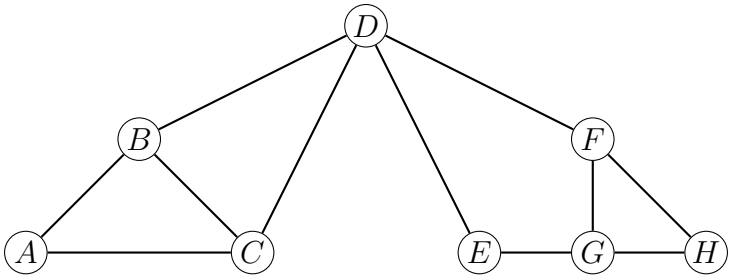
(a) [4 points] Suppose that you want to find a path from A to H. If you use breadth-first search, write down the resulting path as a sequence of vertices.

(b) [4 points] If you use depth-first search to find a path from A to H, write down the resulting path as a sequence of vertices.

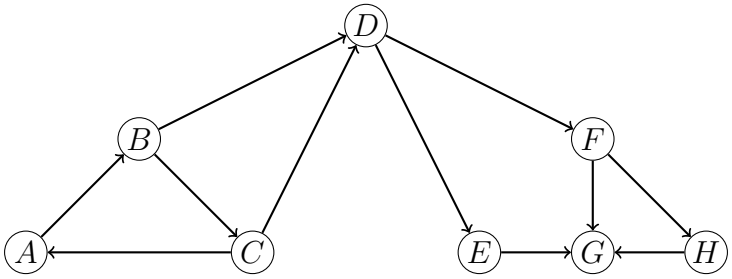
(c) [6 points] To determine whether the maze has cycles or multiple paths to the same destination, you decide to use the edge classification of depth-first search. Run depth-first search on the graph reproduced below, starting from vertex A, and label every edge with T if it's a tree edge, B if it's a back edge, F if it's a forward edge, and C if it's a cross edge.

As a reminder, recall that an edge (u, v) is

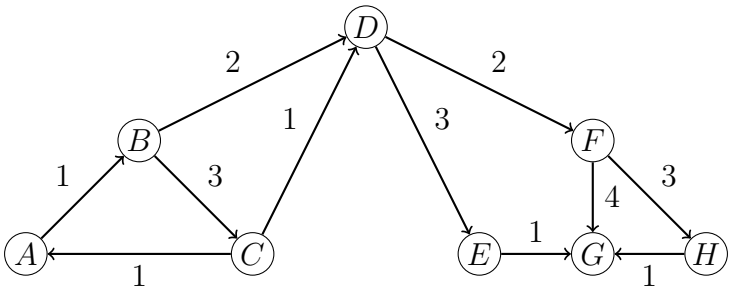
- a ***tree edge*** (T) if v was first discovered by exploring edge (u, v) (tree edges form the depth-first forest);
- a ***back edge*** (B) if v is u's ancestor in a depth-first tree;
- a ***forward edge*** (F) if v is u's descendant in a depth-first tree; and
- a ***cross edge*** (C) if none of the above apply.



(d) [6 points] Now suppose that the passages in the maze are directional. Rerun depth-first search in the directed graph below and label the edges accordingly.



(e) [6 points] Suppose each passage in the maze causes a different amount of ***damage*** to you in game. You change the graph to use weights to represent the damage caused by each edge. You then use Dijkstra's algorithm to find the path from A to H with the lowest possible damage. Write down the order in which vertices get removed from the priority queue when running Dijkstra's algorithm.



Problem 4. Malfunctioning Calculator [25 points] (5 parts)

Former 6.006 student Mallory Funke is at a math competition, but her calculator isn't working. It seems to work fine for whole numbers, but the numbers after the decimal point always seem to be the sequence 07734 repeated over and over again, making those digits useless. For one of the problems, she has been asked to compute $\lfloor A^{3/4} \rfloor$ for a few different integer values of A. Mal knows that Newton's Method can be used to compute the square root or the cube root of an integer A. So as a first step in computing $\lfloor A^{3/4} \rfloor$, Mal wants to use Newton's Method to compute $\lfloor A^{1/4} \rfloor$. She then plans to use that information to compute $\lfloor A^{3/4} \rfloor$.

(a) [5 points] Mal decides to use the function $f(x) = x^4 - A$, because it has a root at $x = A^{1/4}$. Use Newton's Method on $f(x)$ to generate a formula for computing increasingly accurate estimates of $\lfloor A^{1/4} \rfloor$. In other words, give a formula for the more accurate estimate x_{i+1} in terms of a less accurate estimate x_i. The formula you construct must use **only** addition, subtraction, multiplication, and division. (You do not need to simplify the formula.)

(b) [5 points] Mal decides to use the technique from part **(a)** to compute the value $B = \lfloor A^{1/4} \rfloor$. She then plans to compute $\lfloor A^{3/4} \rfloor$ by calculating the value $C = B^3 = B \cdot B \cdot B$. Provide an explanation of why this technique does not work.

Hint: Define α to be the fractional part of $A^{1/4}$, so that $B = A^{1/4} - \alpha$. What happens when you compute $C = B^3$?

(c) [5 points] Mal clearly needs a way to check her answer for $\lfloor A^{3/4} \rfloor$, using only integers. Given a pair of positive integers A and C, explain how to check whether $C \leq A^{3/4}$ using $O(1)$ additions, subtractions, multiplications, and comparisons.

(d) [5 points] Explain how to check whether $C = \lfloor A^{3/4} \rfloor$, again using only $O(1)$ additions, subtractions, multiplications, and comparisons.

Hint: Recall how the floor function is defined.

(e) [5 points] Give a brief description of an algorithm that takes as input a d-digit positive integer A and computes $\lfloor A^{3/4} \rfloor$. The only arithmetic operations you can use are integer addition, integer subtraction, integer multiplication, integer division, and integer comparison. Your algorithm should use $\Theta(\lg d)$ arithmetic operations in total, but partial credit will be awarded for using $\Theta(d)$ arithmetic operations. For this question, you may assume that Newton's Method has a quadratic rate of convergence for whatever function you devise.

Problem 5. The Tourist [15 points]

Your new startup, *Bird Tours*, brings people around Boston in a new aviation car that can both drive and fly. You've constructed a weighted directed graph $G = (V, E, w)$ representing the best time to drive or fly between various city sites (represented by vertices in V). You've also written a history of Boston, which would be best described by visiting sites $v_0, v_1, \ldots, v_k$ in that order.

Your goal is to find the shortest path in G that visits $v_0, v_1, \ldots, v_k$ in order, possibly visiting other vertices in between. (The path must have $v_0, v_1, \ldots, v_k$ as a ***subsequence***; the path is allowed to visit a vertex more than once. For example, $v_0, v_2, v_1, v_2, \ldots, v_k$ is legal.) To do the computation, you've found an online service, Paths 'Я Us, that will compute the shortest path from a given source s to a given target t in a given weighted graph, for the bargain price of \$1. You see how to solve the problem by paying \$$k$, calling Paths 'Я Us with $(v_0, v_1, G), (v_1, v_2, G), \ldots, (v_{k-1}, v_k, G)$ and piecing together the paths. Describe how to solve the problem with only \$1 by calling Paths 'Я Us with (s, t, G') for a newly constructed graph $G' = (V', E', w')$, and converting the resulting path into a path in G.

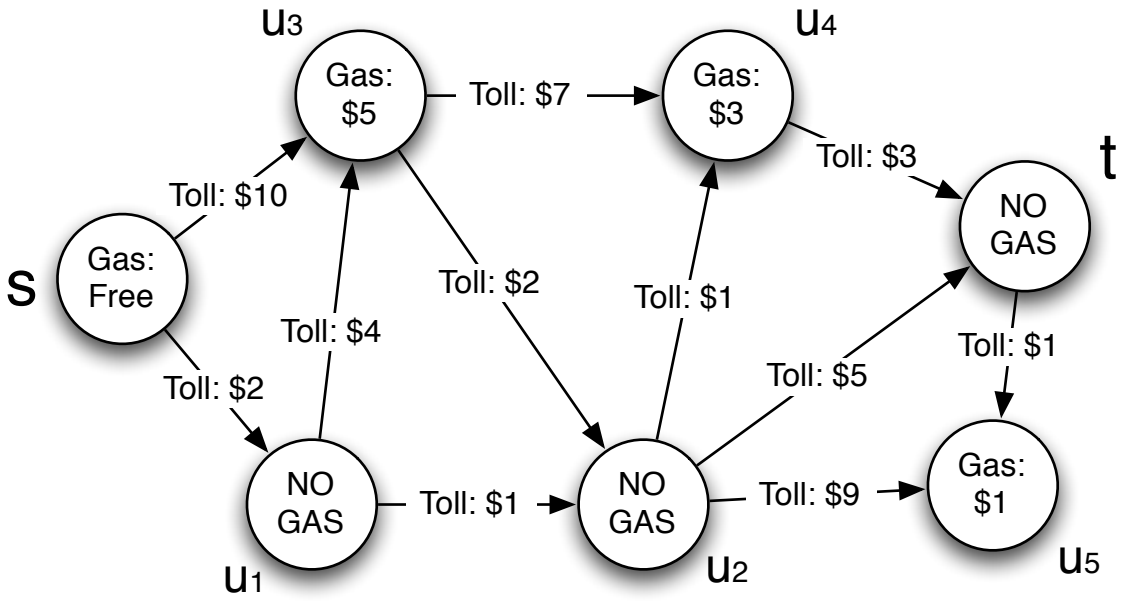
Problem 6. Fill 'Er Up! [25 points]

You are traveling by car from one city to another city. Unfortunately, you have a hole in your gas tank, and you have to refill your gas tank to travel across more than two roads. In addition, there is a toll booth on every road that charges you for using that road. Your goal is to find the least-expensive path from your start to your destination.

You represent the city network using a directed graph $G = (V, E, w)$ with weights w defined on both edges and vertices. The vertices V represent the cities and the edges E represent the roads. The weight $w(e)$ of an edge e represents the toll amount on that road. The weight $w(v)$ of a vertex v is the price of filling your gas tank in that city (which is a fixed price independent of how much gas you have left, or ∞ if there is no gas available to purchase). You are allowed (but not obligated) to end your journey with an empty tank, and you may assume that you always start your journey with a full tank.

Below is an example graph that we will use to answer part (a). One seemingly cheap path from s to t is (s, u_1, u_2, t) at a cost of \$8. Unfortunately, this path is not valid, because our leaky gas tank won't permit moving across three edges without refilling our gas tank.

One valid path is (s, u_3, u_2, t) at a cost of \$22. (This is a valid path: we begin with a full tank, travel across one edge to a gas station where we refill our tank, and then travel two edges to the destination, arriving with an empty gas tank. Notice that we are unable to continue the journey to u_5 if we wanted to, because even though there is a gas station there, we have to traverse a third edge to get to it.)



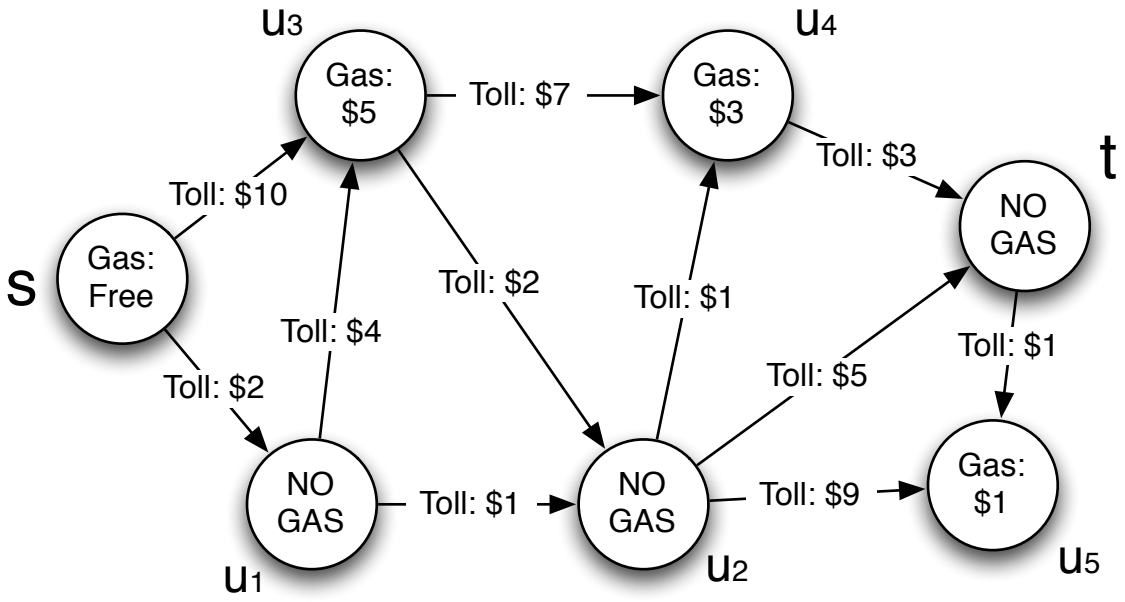
There are some extra copies of this graph at the end of the exam (for your convenience).

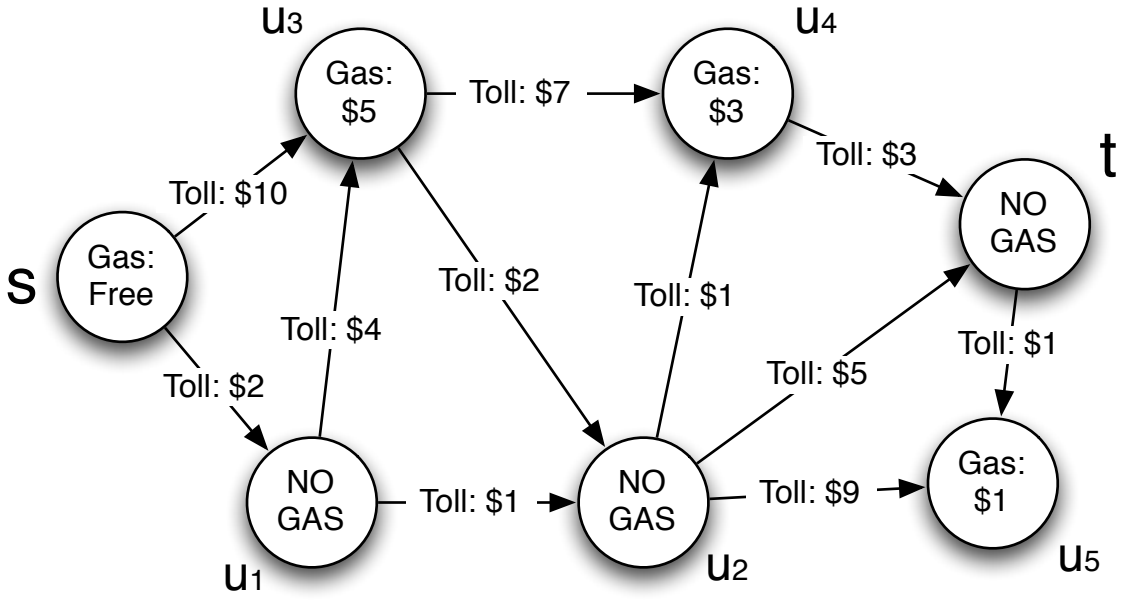
(a) [5 points] The valid path given in the description above is not the *best* path. Find a least-expensive path from s to t in the graph above.

(b) [5 points] Find the least-expensive path from s to u_5 in the graph above.

(c) [15 points] Give an $O(V \log V + E)$ algorithm to find, in a given graph $G = (V, E, w)$, a least-expensive *valid* path from a city s to a city t, or report that no such path exists.

Here are some extra copies of the graph from Problem 6 (for your convenience).





6.006 Introduction to Algorithms
Fall 2011

2

Subject: Quiz 2

Year: Month: Date:

Problem 1 Behyamin Tazayeri

Problem 2

(a) T ✓

(b) F ✓

(c) ~~F~~ F ✓

(d) F ✓

(e) F ✓

(f) F ✗

(g) F ✗

(h) F ✓

(i) T ✗

(j) F ✓

(k) ~~T~~ T ✓

(l) T ✗

(m) F ✓

(n) T ✓

$\frac{102}{120} = 85\%$

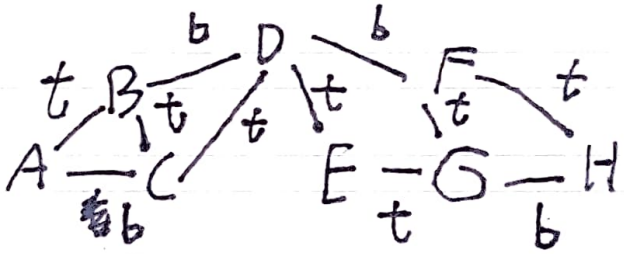
Subject: Year: Month: Date:

Problem 3

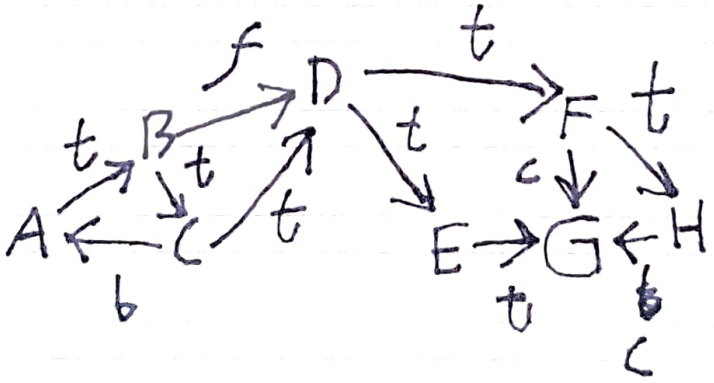
(a) A - B - D - F - H ✓ - F

(b) A - B - C - D - E - G - H ✓

(c)



(d)



(e) ~~A B D C F E G H~~ A B D C F E G H ✓

Subject:

Problem 4

(a) $x_{i+1} = x_i - \frac{x_i^4 - A}{4x_i^3}$ ✓

(b)

$$C = (A^{1/4} - \alpha)(A^{1/4} - \alpha)(A^{1/4} - \alpha)$$ ✓

(c) We do $C^4 \leq A^3$ ✓

(d) $x = 5$

(e) $B = A^3$ ✓

Newton: $f(x) = x^4 - B$

Problem 5

make K copies of G ✓

Connect ~~each layer~~ first and second V_0

2nd & 3rd V_1 etc...

find P between $V_{0,1}$ and $V_{K,K}$

Subject:

Year: Month: Date:

Problem 6

(a) $S - u_1 - u_3 - u_2 - t$ 2. $

(b) $S - u_1 - u_3 - u_2 - t - u_5$ 21 $

(c) We use implicit rep.

At each Gas we BFS up to level 2

copy the induced sub graph of the visited nodes and join at gas.

then run Dijkstra. 10

-5

Final Exam

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 180 minutes to earn 180 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- This quiz is closed book. You may use **three** $8\frac{1}{2}'' \times 11''$ or A4 or 6.006 cushions as crib sheets (both sides). No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- **When writing an algorithm, a *clear* description in English will suffice. Pseudo-code is not required.**
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader		Problem	Parts	Points	Grade	Grader
1	18	36				6	2	20		
2	3	9				7	5	15		
3	5	20				8	6	20		
4	3	15				9	1	15		
5	1	10				10	5	20		
						Total		180		

Name: _____

Wed/Fri Recitation:	Ying	Kevin	Sarah	Yafim	Victor
	10, 11 AM	**11 AM**	**12, 1 PM**	**12 PM**	**2, 3 PM**

Problem 1. True/False [36 points] (18 parts)

Circle (T)rue or (F)alse. You don't need to justify your choice.

(a) T F [2 points] Polynomial: good. Exponential: bad.

(b) T F [2 points] Radix sort runs correctly when using any correct sorting algorithm to sort each digit.

(c) T F [2 points] Given an array $A[1..n]$ of integers, the running time of Counting Sort is polynomial in the input size n.

(d) T F [2 points] Given an array $A[1..n]$ of integers, the running time of Heap Sort is polynomial in the input size n.

(e) T F [2 points] Any n-node unbalanced tree can be balanced using $O(\log n)$ rotations.

(f) T F [2 points] If we augment an n-node AVL tree to store the size of every rooted subtree, then in $O(\log n)$ we can solve a *range query*: given two keys x and y, how many keys are in the interval $[x, y]$?

(g) T F [2 points] AVL trees can be used to implement an optimal comparison-based sorting algorithm.

(h) T F [2 points] Given a connected graph $G = (V, E)$, if a vertex $v \in V$ is visited during level k of a breadth-first search from source vertex $s \in V$, then every path from s to v has length at most k.

(i) T F [2 points] Depth-first search will take $\Theta(V^2)$ time on a graph $G = (V, E)$ represented as an adjacency matrix.

(j) T F [2 points] Given an adjacency-list representation of a directed graph $G = (V, E)$, it takes $O(V)$ time to compute the in-degree of every vertex.

(k) T F [2 points] For a dynamic programming algorithm, computing all values in a bottom-up fashion is asymptotically faster than using recursion and memoization.

(l) T F [2 points] The running time of a dynamic programming algorithm is always $\Theta(P)$ where P is the number of subproblems.

(m) T F [2 points] When a recurrence relation has a cyclic dependency, it is impossible to use that recurrence relation (unmodified) in a correct dynamic program.

(n) T F [2 points] For every dynamic program, we can assign weights to edges in the directed acyclic graph of dependences among subproblems, such that finding a shortest path in this DAG is equivalent to solving the dynamic program.

(o) T F [2 points] Every problem in NP can be solved in exponential time.

(p) T F [2 points] If a problem X can be reduced to a known NP-hard problem, then X must be NP-hard.

(q) T F [2 points] If P equals NP, then NP equals NP-complete.

(r) T F [2 points] The following problem is in NP: given an integer $n = p \cdot q$, where p and q are N-bit prime numbers, find p or q.

Problem 2. Sorting Scenarios [9 points] (3 parts)

Circle the number next to the sorting algorithm covered in 6.006 that would be the best (i.e., most efficient) for each scenario in order to reduce the expected running time. You do not need to justify your answer.

(a) [3 points] You are running a library catalog. You know that the books in your collection are almost in sorted ascending order by title, with the exception of one book which is in the wrong place. You want the catalog to be completely sorted in ascending order.

1. Insertion Sort
2. Merge Sort
3. Radix Sort
4. Heap Sort
5. Counting Sort

(b) [3 points] You are working on an embedded device (an ATM) that only has 4KB (4,096 bytes) of free memory, and you wish to sort the 2,000,000 transactions withdrawal history by the amount of money withdrawn (discarding the original order of transactions).

1. Insertion Sort
2. Merge Sort
3. Radix Sort
4. Heap Sort
5. Counting Sort

(c) [3 points] To determine which of your Facebook friends were early adopters, you decide to sort them by their Facebook account ids, which are 64-bit integers. (Recall that you are super popular, so you have very many Facebook friends.)

1. Insertion Sort
2. Merge Sort
3. Radix Sort
4. Heap Sort
5. Counting Sort

Problem 3. Hotel California [20 points] (5 parts)

You have decided to run off to Los Angeles for the summer and start a new life as a rockstar. However, things aren't going great, so you're consulting for a hotel on the side. This hotel has N one-bed rooms, and guests check in and out throughout the day. When a guest checks in, they ask for a room whose number is in the range $[l, h]$.[1]

You want to implement a data structure that supports the following data operations as efficiently as possible.

1. INIT(N): Initialize the data structure for N empty rooms numbered $1, 2, \ldots, N$, in polynomial time.
2. COUNT(l, h): Return the number of **available** rooms in $[l, h]$, in $O(\log N)$ time.
3. CHECKIN(l, h): In $O(\log N)$ time, return the first empty room in $[l, h]$ and mark it occupied, or return NIL if all the rooms in $[l, h]$ are occupied.
4. CHECKOUT(x): Mark room x as not occupied, in $O(\log N)$ time.

(a) [6 points] Describe the data structure that you will use, and any invariants that your algorithms need to maintain. You may use any data structure that was described in a 6.006 lecture, recitation, or problem set. Don't give algorithms for the operations of your data structure here; write them in parts (b)–(e) below.

[1] Conferences often reserve a contiguous block of rooms, and attendees want to stay next to people with similar interests.

(b) [3 points] Give an algorithm that implements INIT(N). The running time should be polynomial in N.

(c) [3 points] Give an algorithm that implements COUNT(l, h) in $O(\log N)$ time.

(d) [5 points] Give an algorithm that implements CHECKIN(l, h) in $O(\log N)$ time.

(e) [3 points] Give an algorithm that implements CHECKOUT(x) in $O(\log N)$ time.

Problem 4. Hashing [15 points] (3 parts)

Suppose you combine open addressing with a limited form of chaining. You build an array with m slots that can store two keys in each slot. Suppose that you have already inserted n keys using the following algorithm:

1. Hash (key, probe number) to one of the m slots.
2. If the slot has less than two keys, insert it there.
3. Otherwise, increment the probe number and go to step 1.

Given the resulting table of n keys, we want to insert another key. We wish to compute the probability that the first probe will successfully insert this key, i.e., the probability that the first probe hits a slot that is either completely empty (no keys stored in it) or half-empty (one key stored in it).

You can make the uniform hashing assumption for all the parts of this question.

(a) [5 points] Assume that there are exactly k slots in the table that are completely full. What is the probability $s(k)$ that the first probe is successful, given that there are exactly k full slots?

(b) [5 points] Assume that $p(k)$ is the probability that there are exactly k slots in the table that are completely full, given that there are already n keys in the table. What is the probability that the first probe is successful in terms of $p(k)$?

(c) [5 points] Give a formula for $p(0)$ in terms of m and n.

Problem 5. The Quadratic Method [10 points] (1 parts)

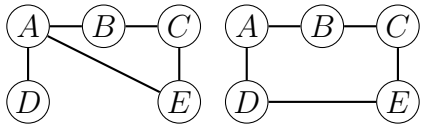
Describe how you can use Newton's method to find a root of $x^2 + 4x + 1 = 0$ to d digits of precision. Either reduce the problem to a problem you have already seen how to solve in lecture or recitation, or give the formula for one step of Newton's method.

Problem 6. The Wedding Planner [20 points] (2 parts)

You are planning the seating arrangement for a wedding given a list of guests, V.

(a) [10 points] Suppose you are also given a lookup table T where $T[u]$ for $u \in V$ is a list of guests that u knows. If u knows v, then v knows u. You are required to arrange the seating such that any guest at a table knows every other guest sitting at the same table either directly or through some other guests sitting at the same table. For example, if x knows y, and y knows z, then x, y, z can sit at the same table. Describe an efficient algorithm that, given V and T, returns the minimum number of tables needed to achieve this requirement. Analyze the running time of your algorithm.

(b) [10 points] Now suppose that there are only two tables, and you are given a different lookup table S where $S[u]$ for $u \in V$ is a list of guests who are on bad terms with u. If v is on bad terms with u, then u is on bad terms with v. Your goal is to arrange the seating such that no pair of guests sitting at the same table are on bad terms with each other. Figure 1 below shows two graphs in which we present each guest as a vertex and an edge between two vertices means these two guests are on bad terms with each other. Figure 1(a) is an example where we can achieve the goal by having A, C sitting at one table and B, E, D sitting at another table. Figure 1(b) is an example where we cannot achieve the goal. Describe an efficient algorithm that, given V and S, returns TRUE if you can achieve the goal or FALSE otherwise. Analyze the running time of your algorithm.



(a) (b)

Figure 1: Examples of guest relationships presented as graphs.

Problem 7. How Fast Is Your Dynamic Program? [15 points] (5 parts)

In the dynamic programs below, assume the input consists of an integer S and a sequence $x_0, x_1, \ldots, x_{n-1}$ of integers between 0 and S. Assume that each dynamic program uses subproblems (i, X) for $0 \leq i < n$ and $0 \leq X \leq S$ (just like Knapsack). Assume that the goal is to compute $DP(0, S)$, and that the base case is $DP(n, X) = 0$ for all X. **Assume that the dynamic program is a memoized recursive algorithm, so that only needed subproblems get computed.** Circle the number next to the correct running time for each dynamic program.

(a) $$DP(i, X) = \max \left\{ \begin{array}{ll} DP(i+1, X) + x_i, & \\ DP(i+1, X - x_i) + x_i^2 & \text{if } X \geq x_i \end{array} \right\}$$

1. Exponential
2. Polynomial
3. Pseudo-polynomial
4. Infinite

(b) $$DP(i, X) = \max \left\{ \begin{array}{ll} DP(i+1, S) + x_i, & \\ DP(0, X - x_i) + x_i^2 & \text{if } X \geq x_i \end{array} \right\}$$

1. Exponential
2. Polynomial
3. Pseudo-polynomial
4. Infinite

(c) $\boxed{DP(i, X) = \max \left\{ \begin{array}{l} DP(i+1, 0) + x_i, \\ DP(0, X - x_i) + x_i^2 \quad \text{if } X \geq x_i \end{array} \right\}}$

1. Exponential
2. Polynomial
3. Pseudo-polynomial
4. Infinite

(d) $\boxed{DP(i, X) = \max \left\{ \begin{array}{l} DP(i+1, X) + x_i, \\ DP(i+1, 0) + x_i^2 \end{array} \right\}}$

1. Exponential
2. Polynomial
3. Pseudo-polynomial
4. Infinite

(e) $\boxed{DP(i, X) = \max \left\{ \begin{array}{l} DP(i+1, X - \sum S) + (\sum S)^2 \\ \text{for every subset } S \subseteq \{x_0, x_1, \ldots, x_{n-1}\} \end{array} \right\}}$

1. Exponential
2. Polynomial
3. Pseudo-polynomial
4. Infinite

Problem 8. Longest Alternating Subsequence [20 points] (6 parts)

Call a sequence $y_1, y_2, \ldots, y_n$ ***alternating*** if every adjacent triple y_i, y_{i+1}, y_{i+2} has either $y_i < y_{i+1} > y_{i+2}$, or $y_i > y_{i+1} < y_{i+2}$. In other words, if the sequence increased between y_i and y_{i+1}, then it should then decrease between y_{i+1} and y_{i+2}, and vice versa.

Our goal is to design a dynamic program that, given a sequence $x_1, x_2, \ldots, x_n$, computes the length of the longest alternating subsequence of $x_1, x_2, \ldots, x_n$. The subproblems we will use are prefixes, augmented with extra information about whether the longest subsequence ends on a descending pair or an ascending pair. In other words, the value $DP(i, b)$ should be the length of the longest alternating subsequence that ends with x_i, and ends in an ascending pair if and only if b is TRUE.

For the purposes of this problem, we define a length-one subsequence to be both increasing and decreasing at the end.

For example, suppose that we have the following sequence:

$$x_1 = 13 \qquad x_2 = 93 \qquad x_3 = 86 \qquad x_4 = 50 \qquad x_5 = 63 \qquad x_6 = 4$$

Then $DP(5, \text{TRUE}) = 4$, because the longest possible alternating sequence ending in x_5 with an increase at the end is is x_1, x_2, x_4, x_5 or x_1, x_3, x_4, x_5. However, $DP(5, \text{FALSE}) = 3$, because if the sequence has to decrease at the end, then x_4 cannot be used.

(a) [4 points] Compute all values of $DP(i, b)$ for the above sequence. Place your answers in the following table:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$b =$ TRUE						
$b =$ FALSE						

(b) [4 points] Give a recurrence relation to compute $DP(i, b)$.

(c) [4 points] Give the base cases of your recurrence relation.

(d) [3 points] Give a valid ordering of subproblems for a bottom-up computation.

(e) [3 points] If you were given the values of $DP(i, b)$ for all $1 \leq i \leq n$ and all $b \in \{\text{TRUE}, \text{FALSE}\}$, how could you use those values to compute the length of the longest alternating subsequence of $x_1, x_2, \ldots, x_n$?

(f) [2 points] When combined, parts **(b)** through **(e)** can be used to write an algorithm such as the following:

LONGESTALTERNATINGSUBSEQUENCE$(x_1, \ldots, x_n)$

```
1   initialize table T
2   for each subproblem (i, b), in the order given by part (d)
3       if (i, b) is a base case
4           use part (c) to compute DP(i, b)
5       else
6           use part (b) to compute DP(i, b)
7
8       store the computed value of DP(i, b) in the table T
9
10  use part (e) to find the length of the overall longest subsequence
```

Analyze the running time of this algorithm, given your answers to the questions above.

Problem 9. Paren Puzzle [15 points]

Your local school newspaper, *The TEX*, has started publishing puzzles of the following form:

Parenthesize $6 + 0 \cdot 6$ to maximize the outcome.

Wrong answer: $6 + (0 \cdot 6) = 6 + 0 = 6$.
Right answer: $(6 + 0) \cdot 6 = 6 \cdot 6 = 36$.

Parenthesize $0.1 \cdot 0.1 + 0.1$ to maximize the outcome.

Wrong answer: $0.1 \cdot (0.1 + 0.1) = 0.1 \cdot 0.2 = 0.02$.
Right answer: $(0.1 \cdot 0.1) + 0.1 = 0.01 + 0.1 = 0.11$.

To save yourself from tedium, but still impress your friends, you decide to implement an algorithm to solve these puzzles. The input to your algorithm is a sequence $x_0, o_0, x_1, o_1, \ldots, x_{n-1}, o_{n-1}, x_n$ of $n + 1$ real numbers $x_0, x_1, \ldots, x_n$ and n operators $o_0, o_1, \ldots, o_{n-1}$. Each operator o_i is either addition ($+$) or multiplication ($\cdot$). Give a polynomial-time dynamic program for finding the optimal (maximum-outcome) parenthesization of the given expression, and analyze the running time.

Problem 10. Sorting Fluff [20 points] (5 parts)

In your latest dream, you find yourself in a prison in the sky. In order to be released, you must order N balls of fluff according to their weights. Fluff is really light, so weighing the balls requires great care. Your prison cell has the following instruments:

- A magic balance scale with 3 pans. When given 3 balls of fluff, the scale will point out the ball with the median weight. The scale only works reliably when each pan has exactly 1 ball of fluff in it. Let $\text{MEDIAN}(x, y, z)$ be the result of weighing balls x, y and z, which is the ball with the median weight. If $\text{MEDIAN}(x, y, z) = y$, that means that either $x < y < z$ or $z < y < x$.
- A high-precision classical balance scale. This scale takes 2 balls of fluff, and points out which ball is lighter; however, because fluff is very light, the scale can only distinguish between the overall lightest and the overall heaviest balls of fluff. Comparing any other balls will not yield reliable results. Let $\text{LIGHTEST}(a, b)$ be the result of weighing balls a and b. If a is the lightest ball and b is the heaviest ball, $\text{LIGHTEST}(a, b) = a$. Conversely, if a is the heaviest ball and b is the lightest ball, $\text{LIGHTEST}(a, b) = b$. Otherwise, $\text{LIGHTEST}(a, b)$'s return value is unreliable.

On the bright side, you can assume that all N balls have different weights. Naturally, you want to sort the balls using as few weighings as possible, so you can escape your dream quickly and wake up before 4:30pm!

To ponder this challenge, you take a nap and enter a second dream within your first dream. In the second dream, a fairy shows you the lightest and the heaviest balls of fluff, but she doesn't tell you which is which.

(a) [2 points] Give a quick example to argue that you cannot use MEDIAN alone to distinguish between the lightest and the heaviest ball, but that LIGHTEST can let you distinguish.

(b) [4 points] Given l, the lightest ball l pointed out by the fairy, use $O(1)$ calls to MEDIAN to implement LIGHTER(a, b), which returns TRUE if ball a is lighter than ball b, and FALSE otherwise.

After waking up from your second dream and returning to the first dream, you realize that there is no fairy. Solve the problem parts below without the information that the fairy would have given you.

(c) [6 points] Give an algorithm that uses $O(N)$ calls to MEDIAN to find the heaviest and lightest balls of fluff, without identifying which is the heaviest and which is the lightest.

(d) [2 points] Explain how the previous parts should be put together to sort the N balls of fluff using $O(N \log N)$ calls to MEDIAN and $O(1)$ calls to LIGHTEST.

(e) [6 points] Argue that you need at least $\Omega(N \log N)$ calls to MEDIAN to sort the N fluff balls.

6.006 Introduction to Algorithms
Fall 2011

Final Exam

Subject: [illegible]

Year: Month: Date:

Problem 0 Benyamin Jazayeri

Problem 1

(a) T ✓

(b) F ✓

(c) T ✓

(d) T ✓

(e) F ✓

(f) T ✓

(g) T ✓

(h) F ✓

(i) T ✓

(j) F ✓

(k) F ✓

(l) F ✓

(m) T ✓

(n) F ✓

(o) T ✓

(p) F ✓

(q) F × -2

(r) T ✓

160/180 = 88.9%

Subject: Year: Month: Date:

Problem 2

(a) 1 ✓

(b) ~~3~~ -3

(c) 3 ✓

Problem 3

(a) we will use an AVL Tree and augment it with satellite data of the size of the subtree rooted at a node ✓ this is our extra invariant.

(b) We need not do anything. -2

when room i gets occupied.

we insert i. when out, we delete it.

(c) we find L or Suc(L) and h or Pred(h)

we find their LCA

then LCA - L.size - h.size + 2 Conditional

Subject: Year: Month: Date:

(d) if Suc(L) or Pred(h) in (L,h) is True

else: Nil (e) Del(i) -1

Problem 4

(a) $\frac{m-k}{m}$

(b) $\sum_{k=0}^{m/2} \frac{m-k}{m} \cdot P(k)$ -2

(c) $\frac{\binom{m}{n}}{m^n}$

Problem 5

$x^2+4x+1 = (x+2)^2 - 3 = 0$

So $x+2 = \sqrt{3}$ So $x = \sqrt{3} - 2$

we use newton to find $\sqrt{3}$ then

Subject:

Year: Month: Date:

Problem 6

(a) We sort V by degree. deleting the node with the biggest degree and its neighbors. return the # of deletions. O(V) -2

(b) we need to check if bipartite. run dfs O(V+E). Check cycle length. if odd False else True.

Subject: Year: Month: Date:

Problem 7

(a) 3 ✓

(b) 4 ✓

(c) 4 ✗ -> 3

(d) 3 ✗ -> 2

(e) 1 ✓

Problem 8 ✓

(a)

	i=1	i=2	i=3	i=4	i=5	i=6
b=True	1	2	2	2	4	4
b=False	1	1	3	3	3	5

(b) $DP(i,b) = DP(i-1, \neg b) + \begin{cases} 1 & \text{if } i == b \\ 0 & \text{else} \end{cases}$ ✓

(c) $DP(0,b) = 1 \cdot \forall b$ - 2

(d) for i 1...n ✓

for j in [b, ¬b]

Subject: Year. Month: Date:

(e) $\max\{DP(i,T), DP(i,F)\}$ ✓

(f) $O(n) - 2$

Problem 9

DP of All Substrings.

Recurrence $\max(DP[i][k] + DP[k][j])$
for $k = i \ldots j$
+ outer ✓

Problem 10.

(a) if we have 3 balls. Excluding the median after that there is nothing to do. ✓

(b) Median (L, a, &b) ✓

(c) ~~Find~~ give some ordering. ✓

Call median(i, i+1, i+2)
for $i = 1 \ldots n-2$

tell med each time.

Subject: Year: Month: Date:

(d) $O(n)$ to find L and h

make < function

$O(n \lg n)$ merge sort.

(e) comparison in tree $N!$ leaves.