CSE 101: Midterm 2

May 3, 2005—Day 14.5

You are allowed a single-sided 8.5x11 page of handwritten notes. Please turn it in along with your test. Use the back pages of the test as scratch paper. If you need additional scratch paper, ask a proctor.

If you need to make any assumptions to solve a problem, *write them down* on the test so that we know what you're thinking.

`Name:` _____

`Student ID:`_____

Problem	Score
1	/4
2	/4
3	/12
4	/10
5	/15
6	/15
7	/20
8	/20
Extra Credit	/10
Total	/100

1. *4 pts.* Assume we have an algorithm A that solves a problem for all instances. When we analyze the runtime, $T(n)$, of A we usually (*choose one*):

   (a) are determining the average runtime across all instances of size n.

   (b) are determining the minimum runtime across all instances of size n.

   (c) are determining the maximum runtime across all instances of size n.

   (d) don't care about the instance size n because we're only interested in asymptotic runtime.

   C) When we analyze runtime, we're interested in worst-case behavior. D) is incorrect: we do care about the size n of the input.

2. *4 pts.* Which of the following is the best description of how Quicksort works? (*Choose one*)

   (a) Divides the problem into two subproblems of equal size and then merges together the results of solving those problems.

   (b) Divides the problem into two subproblems which may or may not be of equal size and then solves those subproblems in-place

   (c) Finds the median of the items and uses that to split the problem into two subproblems that it solves: those items less than the median and those items greater than the median.

   (d) Checks every pair of elements to find those that are out-of-place. When it finds a pair out-of-place, it swaps them.

   B) Quicksort chooses a partition value (splitter) and divides the problem into two subproblems that may or may not be equal size.

   C) Although Quicksort doesn't find the median of the items, it does find the middle item. Given how easily this can be misinterpreted, this answer also counts.

3. *12 pts.* For *each* of the following algorithms, circle which method(s) we used to solve them

   - Sequence Alignment in Linear Space (Greedy, Divide and Conquer, Dynamic Programming)
   - Integer Knapsack (Greedy, Divide and Conquer, Dynamic Programming)
   - Equal-weight Interval Scheduling (Greedy, Divide and Conquer, Dynamic Programming)
   - Weighted Interval Scheduling (Greedy, Divide and Conquer, Dynamic Programming)
   - Closest Pair of Points (Greedy, Divide and Conquer, Dynamic Programming)
   - Clustering (Greedy, Divide and Conquer, Dynamic Programming)

   Sequence Alignment in Linear Space: Divide and Conquer, Dynamic Programming

   Integer Knapsack: Dynamic Programming

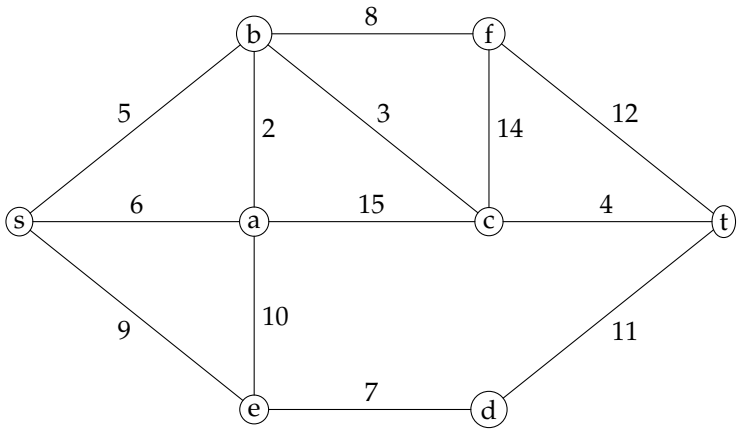
   Equal-weight Interval Scheduling: Greedy

   Weighted Interval Scheduling: Dynamic Programming

   Closest Pair of Points: Divide and Conquer

   Clustering: Greedy

4. *10 pts.* Assume you are using Kruskal's algorithm to find a MST of the following graph. What is the cost of the *fifth* edge that is added to the tree?



The first edge added is ab, second: bc, third: ct, fourth: sb, fifth: de, whose cost is 7.

5. *15pts.* If $T(n) = 6T(n/4) + n^{1.5}$, what is the asymptotic complexity (O, Θ, or Ω notation)? Justify your answer.

   Using the master method, $a = 6, b = 4, f(n) = n^{1.5}$. $n^{\log_4 6} < n^{log_4 8} = n^{1.5}$, so $T(n) = \Theta(n^{1.5})$.

6. *15 points* What is the *exact* number of "Hello"s printed by Proc as a function of n ?

```
function Proc(n)
Precondition: n is a non-negative integer
begin
    print "Hello"; print "Hello"; print "Hello";
     for i = 1 to n do begin
         print "Hello"; print "Hello"; print "Hello"; print "Hello";
         for j = 1 to n do begin
             print "Hello"; print "Hello";
         end
     end
end
```

   The inner loop is executed n^2 times and prints "Hello" twice each loop: $2n^2$. The outer loop is executed n times and prints "Hello" four times each loop: $4n$. Proc prints "Hello" 3 times: 3. Total: $2n^2+4n+3$.

7. *20 pts. total*

Following is an algorithm that correctly creates a Minimal Spanning Tree of a connected graph G by starting with a single node and repeatedly adding the cheapest edge that will expand the tree.

```
Procedure FindMST(E, V)
Precondition: _____________________________________________
_____________________________________________________________
_____________________________________________________________
_____________________________________________________________
Begin
E' = {}
V'= {}
Add one node of V to V'
Loop
    Loop Invariant: _____________________________________________
    _____________________________________________________________
    _____________________________________________________________
    _____________________________________________________________
    _____________________________________________________________
    Exit when |V'| = |V|
    Find least expensive edge e = (x,y), such that x is
        in V' and y is in V - V'
    E' = E' + e;
    V' = V' + {y}
Endloop
End
Postcondition: (E', V') is a MST of (E, V)
```

(a) *10 pts.* Fill in the Precondition and Loop Invariant necessary to prove the Postcondition.

(b) *10 pts.* Prove the algorithm is true by proving the Loop Invariant and Postcondition. Note: The only information you may use about the loop when proving the Postcondition is the Loop Invariant and the Exit Condition.

Precondition: (E, V) is a connected graph with $|V| > 0$.

Loop Invariant: (E′, V′) is an MST of (E, V′); V′ is a subset of V

LI initialization: V′ has only one vertex and E′ is the empty set, so (E′, V′) is an MST of (E, V′). V′ includes one node from V so is a subset of V.

LI maintenance: y is a node from V and V′ starts out a subset of V, so after V″ = V′ + {y}, V″ is a subset of V.

By the cut property, the least expensive edge, e, of a cut between V′ and V-V′ is in any MST of (E,V). Since V″ is a subset of V, e is in the MST of (E, V″).

Termination: Every time through the loop, $|V'|$ increases by one

Proof of postcondition: By the LI, (E′, V′) is an MST of (E, V′). Since V′ is a subset of V, and (by the exit condition) $|V| = |V'|$, V = V′. Thus, (E′, V′) is an MST of (E, V).

8. *20 pts.* You have coins in the following denominations: 1¢, 4¢, 5¢. Use dynamic programming to find what coins to use when giving change for 8¢ (while minimizing the number of coins).

   Make sure to provide: a recurrence, a description of your notation, a filled-out table, and work showing how you're finding which coins to use.

   Define OPT(i) to be the minimum number of coins to give change for i ¢.

   Let OPT(i) = 0 if i $\leq$ 0. Let OPT(i) = 1+Min(OPT(i-5), OPT(i-4), OPT(i-1)).

   That is, if we use a 5¢ coin first, we'll need OPT(i-5) to optimally give change for the remainder. Similarly for using a 4¢ coin first or a 1¢ coin first.

i	OPT(i)
0	0
1	1
2	2
3	3
4	1
5	1
6	2
7	3
8	2

By the table, OPT(8) = 2, so we only need 2 coins to give change. Since OPT(i-4) = OPT(4) = 1, we use a 4¢ coin first. Similarly, since OPT(4-4)=OPT(0) = 0, we use a 4¢ coin second.

9. *Extra credit: 10 points*

   Statement 1: Given an undirected graph, G, with distinct edge costs, let C be any cycle in G, and let edge $e = (v, w)$ be the most expensive edge belonging to C. Then e does not belong to any minimum spanning tree of G.

   Prove or disprove Statement 1.

   Statement 2: Given an undirected graph, G, with distinct edge costs, let C be any cycle in G, and let edge $e = (v, w)$ be the least expensive edge belonging to C. Then e belongs to all minimum spanning tree of G.

   Prove or disprove Statement 2.

   Statement 1 is true. Proof by exchange argument. Assume there's a spanning tree, T, containing e. Consider e=(v,w) in T. Removing (v,w) from T separates T into two trees. Consider the cycle C: it must contain another edge f=(x, y) that connects the two trees since it provides an alternate route between v and w. Let $T' = T + \{f\} = \{e\}$. T' is also a spanning tree, but since f is of lower weight, the total weight of T' is less than that of T, so T was not a MST.

   Statement 2 is false. Counterexample: consider a pentagon with a star embedded in it. Make the weights of the perimeter edges 100, and the weights of the star edges 1. Clearly, the MST of the graph

is the star edges. Now, consider C, the perimeter cycle. None of the 100-weight edges are in the MST.