

*Answer Keys  
for  
Theory of  
Algorithms  
Exams*

*by*

*Peter M. Maurer*

1. Using these formulas:

$$(n+1)^2 = n^2 + 2n + 1$$

$$(n+1)^3 = n^3 + 3n^2 + 3n + 1$$

Prove the following summation formula by induction.  
(15 points).

$$\sum_{i=1}^n (i^2 + 2i + 1) = \frac{2n^3 + 9n^2 + 13n}{6}$$


---

$$\sum_{i=1}^1 (i^2 + 2i + 1) = 1 + 2 + 1 = 4 = \frac{24}{6} = \frac{2 + 22}{6} = \frac{2 + 9 + 13}{6} = \frac{2 \cdot 1^3 + 9 \cdot 1^2 + 13 \cdot 1}{6}$$

$$\sum_{i=1}^{n+1} (i^2 + 2i + 1) = \sum_{i=1}^n (i^2 + 2i + 1) + ((n+1)^2 + 2(n+1) + 1) = \frac{2n^3 + 9n^2 + 13n}{6} + ((n+1)^2 + 2(n+1) + 1) =$$

$$\frac{2n^3 + 9n^2 + 13n}{6} + \frac{6(n^2 + 2n + 1)}{6} + \frac{12(n+1)}{6} + \frac{6}{6} = \frac{2n^3 + 15n^2 + 37n + 24}{6} =$$

$$\frac{2n^3 + 6n^2 + 6n + 2 + 6n^2 + 18n + 6 + 13n + 13}{6} =$$

$$\frac{2(n^3 + 3n^2 + 3n + 1) + 6(n^2 + 2n + 1) + 13(n+1)}{6} = \frac{2(n+1)^3 + 6(n+1)^2 + 13(n+1)}{6}$$

2. Prove that the function  $f(n) = n^k \lg n$  grows faster than  $g(n) = n^k$ , and slower than  $h(n) = n^{k+c}$ , where  $c$  is any **real** number greater than 0.  
(15 points)

---

Assume  $k$  is greater than zero.

$$\lim_{n \rightarrow \infty} \frac{n^k \lg n}{n^k} = \lim_{n \rightarrow \infty} \lg n = \infty$$

$$\lim_{n \rightarrow \infty} \frac{n^{k+c}}{n^k \lg n} = \lim_{n \rightarrow \infty} \frac{n^c}{\lg n} = \lim_{n \rightarrow \infty} \frac{cn^{c-1}}{n^{-1}} = \lim_{n \rightarrow \infty} cn^c = \infty$$

3. The basic operation for the function ABC is the number of times "HELLO WORLD" is printed? What is the time bound of ABC? Give both the recurrence relation that describes the behavior of ABC, and then give the order.  
(20 points)

```

ABC(n)
  if (n > 1) then
    for i := 1 to n do
      j := n;
      while j > 1 do
        PRINT "HELLO WORLD";
        PRINT "HELLO WORLD";
        j := j / 2;
      endfor
    endfor
    for i:= 1 to 8 do
      ABC(n/2);
    endfor
  endif

```

$$T(n) = 8T(\lfloor n/2 \rfloor) + 2\lceil \lg n \rceil \in \Theta(n^3)$$

4. Assuming that  $n=2^k$ , for some integer  $k$ , and that  $T(1) = 0$ , find the exact solution of the following recurrence relation.  
(20 points)

$$T(n) = 4T(n/2) + n^2$$

$$T(n) = 4T(n/2) + n^2 = 4\left(4T(n/4) + \left(\frac{n}{2}\right)^2\right) + n^2 =$$

$$16T(n/4) + n^2 + n^2 = 16\left(4T(n/8) + \left(\frac{n}{4}\right)^2\right) + n^2 + n^2 =$$

$$64T(n/8) + n^2 + n^2 + n^2 = 2^{2i}T(n/2^i) + in^2 =$$

$$2^{2i}T(2^{k-i}) + in^2 = 2^{2k}T(2^{k-k}) + kn^2 = kn^2 = n^2 \lg n$$

5. Suppose you have invented a new version of Quicksort, and you find, much to your surprise, that your new version always selects the second largest element of the list as the pivot point. Prove that both the worst and average case of your new algorithm is  $\Theta(n^2)$ .  
(15 points)

Since the behavior of the algorithm is independent of the input, average and worst case are the same.

Worst case is given by the following recurrence.

$$W(n) = W(n-2) + n - 1 = W(n-4) + n-2 - 1$$

Depending on whether  $n$  is even or odd this is a summation of even or odd numbers of the form

$$\sum_{i=1}^{n/2} 2i \text{ or } \sum_{i=0}^{n/2} (2i+1) \quad \text{The second summation can be obtained from the first by adding } n/2 \text{ to the total.}$$

Therefore, we can obtain a solution of the first summation and add  $n/2$  if it affects the order of the result.

$$\sum_{i=1}^{n/2} 2i = 2 \sum_{i=1}^{n/2} i = 2 \frac{(n/2)((n/2)+1)}{2} = n^2/4 + n/2 \in \Theta(n^2)$$

Adding  $n/2$  will not affect the order of the result.

6. Given a list of 10 or more elements, the problem is to find the second, fifth and seventh largest elements. Give both upper and lower asymptotic bounds on the complexity of this problem. The two bounds need not be equal, but you *must* prove that your bounds are correct. (15 points)

One can sort the list into ascending order, place the result into an array, and simply extract the three desired elements. This will require  $\Theta(n \lg n)$  time, therefore  $\Theta(n \lg n)$  is an upper bound for the problem. To obtain the desired elements, any correct algorithm must examine each element of the list, which will require  $\Theta(n)$  time. Therefore  $\Theta(n)$  is a lower bound for this problem.

# Answer Key

CIS 6930  
Exam #2

Theory of Algorithms  
Open Book, Open notes, 1 hour 15 minute time limit.  
Answer on Separate Sheets

Mar22, 1995  
100 points total.

1. Consider the following variation of Depth First Search.

```
DFS(V:Vertex)
  If V = TargetVertex Then
    Process Parent Array
  Else
    Mark and visit V;
    For every unmarked vertex W adjacent to V do
      Parent[W] = V;
      DFS(W);
    End For;
    Unmark V;
  End If
End DFS
```

Look  
Here!



```
TargetVertex = VertexA;
Parent[VertexB] = 0;
DFS(VertexB);
```

Show (not *too* detailed please) that for any simple path  $P$  between  $VertexA$  and  $VertexB$ , that  $P$  will be processed by the statement “Process Parent Array.” Assume that the size of the input is equal to the number of vertices of the graph to be searched. Show that this algorithm is NOT polynomially bounded. (Hint consider a complete graph on  $n$  vertices.)

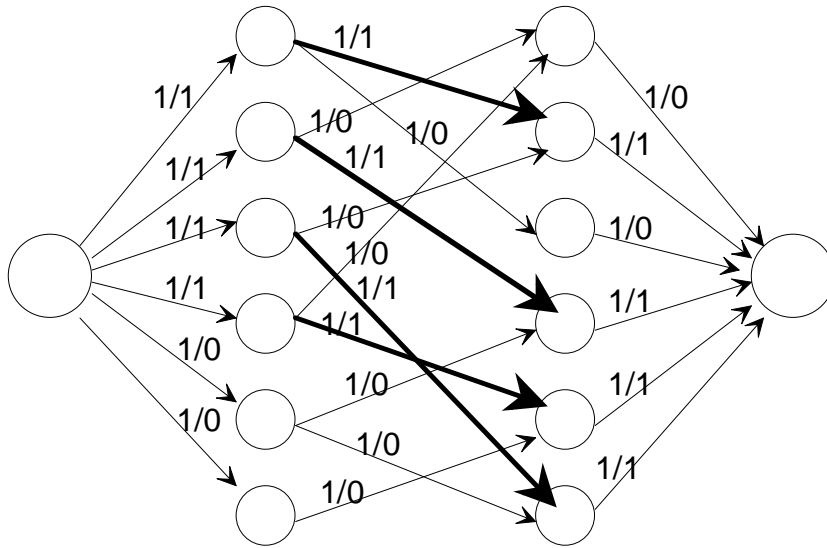
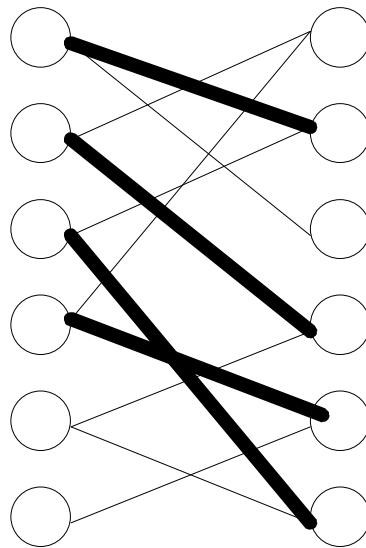
(20 points.)

First observe that the parent array gives a path from the current vertex  $V$  to the root. Next, observe that no vertex is marked unless it is on that path. The *FOR* statement will process every vertex adjacent to  $V$ , except those on the path between  $V$  and the root. To prove the required condition, you must prove something more general, namely that *if there is a simple path of length  $k$  that either does not contain  $VertexB$ , or contains  $VertexB$  as a final element, then that path will be contained in the parent array for some call to DFS.* Proceed by induction. The statement is trivially true if  $k=0$ . Suppose  $k>0$ , and let  $P=V_1, \dots, V_k$  be a path with the required properties. Let  $P'=V_1, \dots, V_{k-1}$ . Since  $P$  is simple,  $P'$  is simple and does not contain  $VertexB$ . Therefore, by the inductive hypothesis,  $P'$  is contained in the Parent array for some call to *DFS*, and by the above observations,  $V_k$  is not marked when this call is made. Because  $V_{k-1}$  does not equal  $VertexB$ , the *DFS* algorithm will enter the *FOR* loop and because  $V_k$  is adjacent to  $V_{k-1}$  and not marked, *DFS* will be called on  $V_k$ . If  $V_k$  happens to equal  $VertexB$ , the parent array will be processed by the statement “Process Parent Array.”

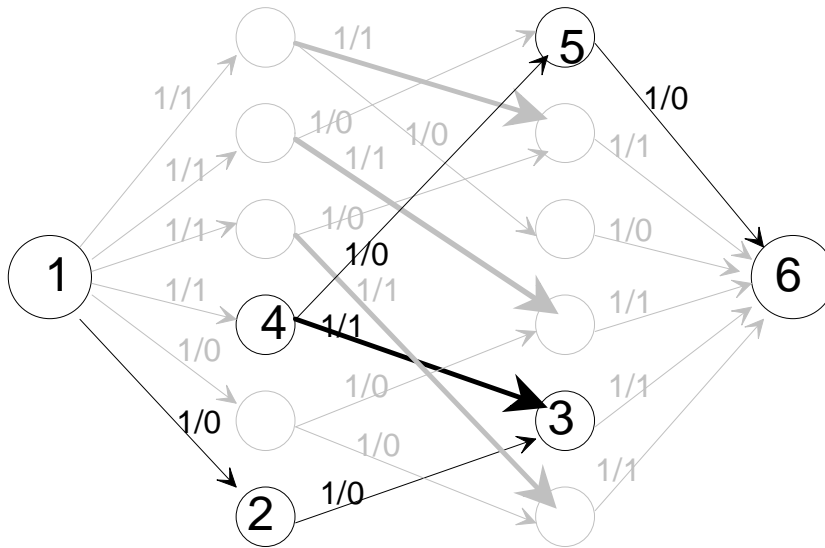
This algorithm is not polynomial. Let  $G$  be a complete graph on  $n$  vertices. Because every vertex is adjacent, every permutation of the vertices of  $G$  is a valid simple path containing  $n$  vertices. In particular, every path of the form  $VertexA, V_1, \dots, V_{n-2}, VertexB$  is a valid simple path between  $VertexA$  and  $VertexB$ . There are  $(n-2)!$  such paths.

# Answer Key

2. Consider the following bipartite matching problem. Show how to convert this to a maximum flow problem. Show a maximum matching. Show an augmenting path containing at least 5 vertices. The fat lines are the existing matches. (20 points)



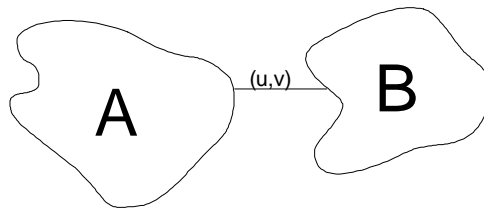
# Answer Key



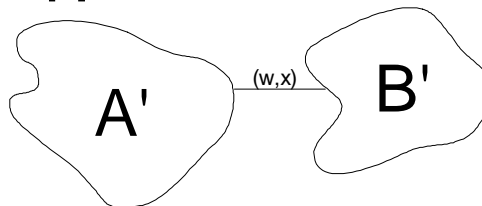
3. Suppose  $S$  and  $T$  are two minimum spanning trees for a graph  $G$ , and that  $S$  and  $T$  are identical except for one edge. In particular  $S$  has the edge  $(u,v)$  which is missing from  $T$ , and  $T$  has the edge  $(w,x)$ , which is missing from  $S$ . Show that  $(u,v)$  and  $(w,x)$  must have the same weight. (20 points)

$S$  and  $T$  look like this:

**S:**

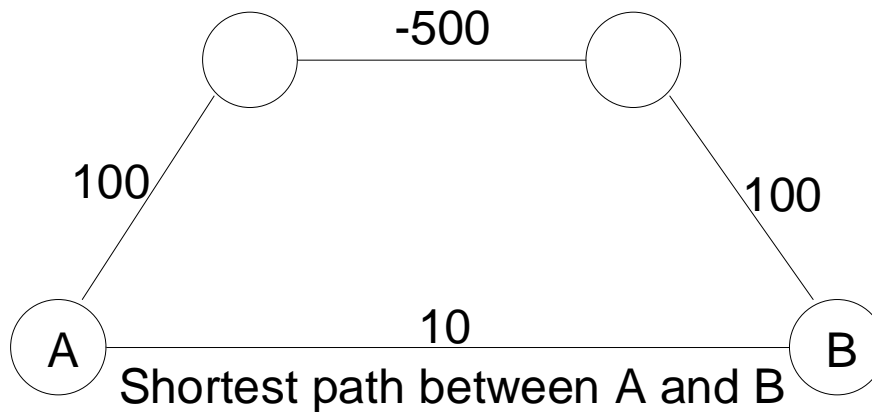


**T:**



Since  $S$  and  $T$  are identical except for the edges  $(u,v)$  and  $(w,x)$ ,  $A=A'$  and  $B=B'$ . Let  $W(A)$  be the weight of all edges in  $A$ , and  $W(B)$  be the weight of all edges in  $B$ . The total weight of  $S$  is  $W(A)+W(B)+W((u,v))$  and the total weight of  $T$  is  $W(A)+W(B)+W((w,x))$ . Since  $S$  and  $T$  are both minimum spanning trees their total weights must be the same, so  $W(A)+W(B)+W((u,v))=W(A)+W(B)+W((w,x))$ . Cancelling  $W(A)+W(B)$  from both sides, we get  $W((u,v))=W((w,x))$ .

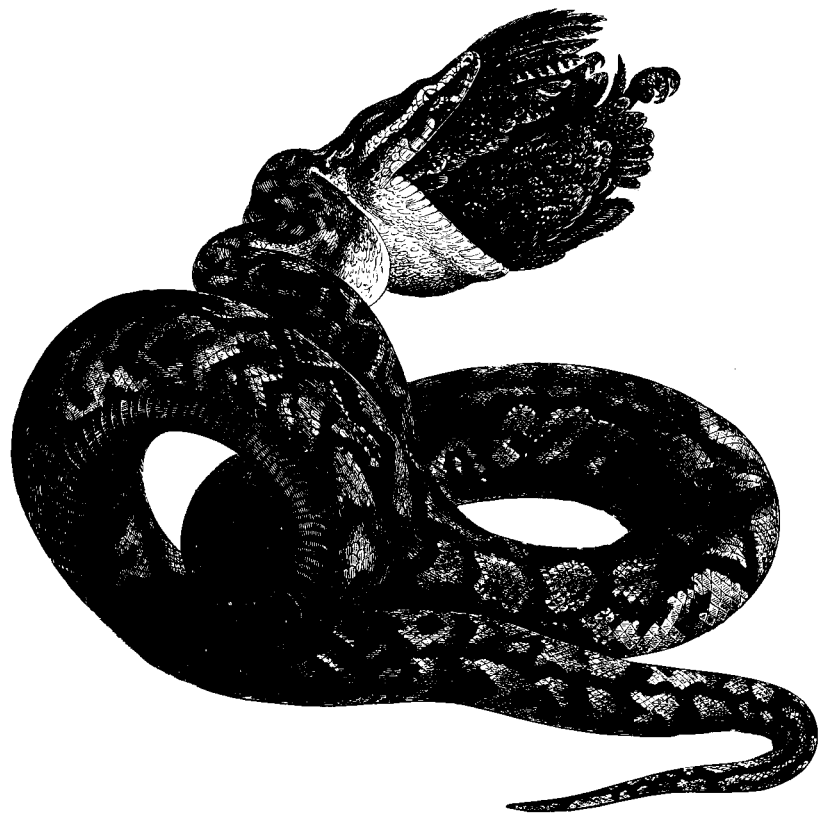
4. Professor Gooms from the University of Florida has told his algorithms class about his wonderful new shortest path algorithm that will work for arbitrary graphs with negative edge weights. “First you find the negative edge weight with the largest absolute value. Call that absolute value  $k$ . Then add  $k+1$  to the weight of each edge. Then find the shortest path using the usual algorithm.” Professor Gooms has blown it (again). Give an example of a graph for which Professor Gooms’ algorithm won’t work.  
(20 points.)



Adding a constant to all weights doesn't change a thing.

5. To show a problem is NP-Complete, you must prove two things. What are they? What does NP-Hard mean?  
(20 points)

To prove that a problem  $X$  is NP-Complete you must show that  $X$  is an element of NP, and that  $X$  is NP-Hard. To say that a problem  $X$  is NP-Hard is to say that any problem  $Y$  which is an element of NP can be reduced to  $X$  in polynomial time. In other words, if  $X$  can be solved in polynomial time then every problem in NP can be solved in polynomial time.



**Complete All Problems on this page.**

1. (20 points) There are *two* things you must prove to show that a problem is NP-Complete. What are they? Explain, in detail, how to do these two things.

To show that problem X is NP-Complete, you must show that

- a. The problem is in NP
- b. The problem is NP-Hard

To show that the problem is in NP, construct a non-deterministic polynomial-time algorithm that solves the problem.

To show that the problem is NP-Hard, show that a known NP-Complete problem can be reduced to X in polynomial time.

2. (20 points) A CNF equation contains the following clauses. What clauses would the equivalent 3-SAT form have?

**{X,Y,Z',W,V}, {Q,X,Y'W'}, {R,Q',T}, {Q,X'} {Z}**

{X,Y,X1}, {X1',Z',X2}, {X2',W,V}, {Q,X,X3},{X3',Y',W'},{R,Q',T},  
 {Q,X',X4},{Q,X',X4'},{Z,X5,X6},{Z,X5',X6},{Z,X5,X6'},{Z,X5',X6'}

3. (20 Points) Given the following 3-Sat clauses, show the truth setting and satisfaction testing components of the equivalent 3-dimensional matching problem.

**{A,B,C'}, {C',D',A'}, {B',D,C}**

Truth Setting:

(A1,X1,Y1)	(B1,X4,Y4)	(C1,X7,Y7)	(D1,X10,Y10)
(A1',X1,Y2)	(B1',X4,Y5)	(C1',X7,Y8)	(D1',X10,Y11)
(A2,X2,Y2)	(B2,X5,Y5)	(C2,X8,Y8)	(D2,X11,Y11)
(A2',Y2,X3)	(B2',Y5,X6)	(C2',Y8,X9)	(D2',Y11,X12)
(A3,X3,Y3)	(B3,X6,Y6)	(C3,X9,Y9)	(D3,X12,Y12)
(A3',Y3,X1)	(B3',Y6,X4)	(C3',Y9,X7)	(D3',Y12,X10)

Satisfaction Testing

(A1,X13,Y13)	(C2',X14,Y14)	(B3',X15,Y15)
(B1,X13,Y13)	(D2',X14,Y14)	(D3,X15,Y15)
(C1',X13,Y13)	(A2',X14,Y14)	(C3,X15,Y15)

4. (20 points) Give an optimal (in terms of time bound) CREW-PRAM algorithm for finding dot product of two vectors. (Component-wise multiplication, add up products.) How fast does your algorithm run? Could this be used to create a matrix multiplication algorithm? If so, how fast would it run?

```

Read V[i] into x;
Read W[i] into y;
z := x*y;
Write z into M[i];
Write 0 into M[i+n]; {n is vector size}
incr := 1;
for k = 1 to Ceiling(lg n) do
  Read M[i+incr] into x;
  z := z + x;
  Write z into M[i];
  incr := incr * 2;
endfor

```

This algorithm runs in  $\lg n$  time. Using this algorithm on  $n^2$  sets of processors, matrix multiply could be performed in  $\lg n$  time. A total of  $n^3$  processors would be required.

5. (20 points) Assuming that  $N$  is a power of some suitable radix  $a$ , how many times will the line "Hello World" be printed by the procedure Proc1 for the generic argument  $N$ ? What is  $a$ ?

```

procedure Proc1(N:Integer)
begin
  if N <= 1 then
    Print "Hello World"
  else
    for i := 1 to N step 2
      Print "Hello World"
    endfor
    for l := 1 to 4 do
      Proc1(N/9)
    endfor
  endif
end;

```

$$T(1)=1$$

$$T(n)=4T(n/9)+n/2$$

$\log_9 4 < 1$ , so  $T(n) \in \Theta(n)$ ,  $a$  is 9.

**Do as many of these problems as you can.**

6. (20 points) You are working for a bicycle shop, and your employer has recently purchased a large number of chains and sprockets that he wants assembled into bicycles. Unfortunately, the chains and sprockets are all of odd gauges, so they are not interchangeable. After a great deal of effort, you have numbered all the chains and all the sprockets (there are an equal number of each) and you have determined which chains will work with which sprockets. Suggest a method for building the maximum number of bicycles from these parts. Is this problem NP-Complete? If so, suggest a suitable approximation.

Create a bipartite graph where the nodes of type 1 represent chains, the nodes of type 2 represent sprockets, and the edges represent compatibility between a chain and a sprocket. Apply the algorithm for maximum bipartite matching. The problem is NOT NP-Complete.

7. (20 points) Suggest a suitable method for solving the following problem. If it is NP-Complete, suggest a suitable approximation method. In the great Central Iowa ballooning contest the objective is for the winning balloonist to carry as much weight as possible across the finish line. Each balloonist is given a collection of iron blocks of many different sizes. Each balloonist must load as many blocks as possible into the balloon and float across the finish line. Although it would be easy to fit all the blocks into the balloonist's basket, the weight of all the blocks is much too large for any balloon. The problem is to find the maximum weight for each balloon. You may assume that the lifting capacity of each balloon is known, and that an adjustment has already been made for the pilot's weight.

This problem is the subset-sum problem. It is NP-Complete. An exact solution would try every subset, and throw out the subsets whose weight was too heavy. An approximation method is to choose a fixed  $k$ , and try every subset of at least  $k$  elements. For each subset of  $k$  elements, add other elements at random until no more elements can be added without exceeding the maximum weight. Choose the heaviest of the subsets created in this manner.

8. (20 points) Professor Gooms of the University of Florida claims to have discovered a new matrix multiplication method that requires only 3 multiplications and 4 additions to multiply a  $2 \times 2$  matrix. There is something suspicious about this result. What is it? We know that a lower bound for matrix multiply is  $n^2$ . Using the textbook method for computing algorithm run time, we get,  $M(k) = 3M(k-1)$ ,  $k = \lg n$ . But this gives us  $M(n) \in \Theta(n^x)$  where  $x = \lg 3$ .  $\lg 3$  is less than 2.

9. (20 points) Give the worst case time bound for each of the following sort algorithms.

**QuickSort, MergeSort, InsertionSort, HeapSort.**

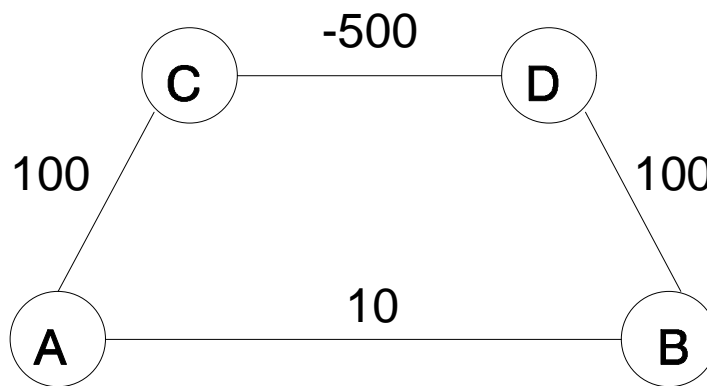
Quicksort -  $n^2$

Mergesort -  $n \lg n$ .

InsertionSort -  $n^2$

HeapSort -  $n \lg n$ .

10. (20 points) Give an example that proves Dijkstra's shortest path algorithm won't work for graphs with negative edge-weights.



Shortest Path from A to B.

1. To show that a problem X is NP complete, you must show that:
  - a.  $X \in NP$
  - b. X is NP-Hard.

To show a, construct a non-deterministic algorithm that solves X in polynomial time.

To show b, transform a known NP Complete problem to X in polynomial time.

2. Assume that  $N^3$  processors are available, and that they are indexed using 3 indices, as in  $P_{i,j,k}$ . Assume that the input is in a pair of two-dimensional arrays A, and B, and that the result will end up in another two-dimensional array C. Assume there is a temporary work area in main memory. This has the form of a three-dimensional array T whose array bounds are N by N by 2N.

```

Read A[i,k] into x;
Read B[k,j] into y;
z = x*y;
Write z into T[i,j,k];
incr = 1
For q = 1 to lg N do
  Read T[i,j,k] into x;
  Read T[i,j,k+incr] into y;
  z = x + y;
  Write z into T[i,j,k];
  incr = incr * 2;
EndFor
if k = 1 then
  Read T[i,j,k] into x;
  Write x into C[i,j];
endif

```

Time bound is  $\lg n$

3. The recurrence relations are
 
$$M(N) = 5M(N/2), \quad M(1) = 1$$

$$A(N) = 5A(N/2) + 12N^2, \quad A(1) = 0$$

These have the solution  $\Theta(N^{\lg 5}) = \Theta(N^{2.32})$

4. Assume that Dijkstra's algorithm could handle negative edge weights. The longest path problem (longest simple path) is known to be NP-complete. Do the following. Start with a graph. If the graph is not weighted, assign a weight of 1 to each edge. Negate the weight of all edges. The longest simple path in the original graph must be the shortest path between some pair of vertices in the new graph. Run Dijkstra's algorithm on every pair of vertices, comparing the total weights of the paths found.

Identify the shortest one as the longest path in the original graph. Since Dijkstra's algorithm constructs only simple paths, and since it runs in polynomial time, the longest path problem has been solved in polynomial time, implying that  $P=NP$ .

5. The recurrence is:

$$T(N) = 0, T < 1$$

$$T(N) = 25T(N/5) + \frac{N^2 + N}{2}$$

By the Master's theorem, we must compare  $\frac{N^2 + N}{2}$  with  $n^x$  where  $x = \log_5 25 = 2$ .

Since the two functions are of the same order, the order of the recurrence is  $\Theta(N^2 \lg N)$

6. This is just the bin packing problem. Weigh all the people and put the heaviest ones in first. This is the Non-Increasing First Fit approximation, which is known to be quite good.

**COT 6505**

**Theory of Algorithms**

**Nov 4, 1998**

**Open Book, Open notes, 1 hour 15 minute time limit**

**Exam #2**

**Answer on Separate Sheets**

**100 points**

1. Let  $G=(V,E)$  be a directed graph. What is the maximum number of high-level calls that must be made when performing a Depth First Search of  $G$ ? Prove your answer. Suggest a method for finding the minimum number of calls.

(10 points.)

**Let  $v$  in  $V$  be a vertex of a strongly connected component  $C$ . Every high-level call with  $v$  will cause all of  $C$  to be visited. Therefore the number of high-level calls cannot exceed the number of strongly connected components of  $G$ . The minimum number of calls is equal to the number of vertices of in-degree 0 in the reduced graph of  $G$ .**

2. In the Strongly Connected Component Algorithm presented in class (Not in the book!) is it possible to use breadth first search instead of depth first search? Prove your answer.

(10 points.)

**No. In breadth first search, it is impossible to distinguish “good” cross edges from “bad” cross edges. Proof requires an example upon which the algorithm fails.**

3. Let  $G$  be a graph with two adjacent vertices  $A$  and  $B$ . Removing the edge  $(A,B)$  causes the graph to become disconnected. Does this mean that  $A$  and  $B$  are articulation points? Prove your answer.

(10 points)

**No. This would be true for any 2-vertex connected graph. Neither vertex is an articulation point.**

4. Let  $G$  be a weighted graph.  $G$  is a simple cycle. Write an algorithm for calculating the number of Minimum Spanning Trees of  $G$ .

(10 points.)

- 1. Find the maximum of the edge weights, assign result to  $m$ .**
- 2. Count the number of edges of weight  $m$ . This is equal to the number of MSTs.**

5. Let  $G$  be a directed graph. Run the following algorithm, AlgorithmX, on  $G$ .

```
AlgorithmX: Begin
  for i:= 1 to n do
    Mark[i] := False;
  endfor
  Mark[j] := True;
  for t := 1 to k do
    for i := 1 to n do
      Ptr := AdjList[i];
      while Ptr = NULL do
        w := Ptr->vertex;
        if Mark[i] then
```

```

        Mark[w] := True;
      endif
      Ptr := Ptr->next;
    endwhile
  endfor
endfor
End

```

Which vertices of  $G$  are guaranteed to be marked when AlgorithmX finishes. Prove your answer. For a specific graph  $G$ , is it possible to guarantee that every vertex of  $G$  will be marked? Prove your answer. What must  $k$  equal if there is to be a chance of marking every vertex of  $G$ ? In class hint: This algorithm is structurally identical to the Bellman-Ford shortest path algorithm.

(10 points)

**Any vertex  $x$  for which there is a path from  $j$  to  $x$  will be marked. No, give an example of a graph for which there is no path from  $j$  to  $x$ .  $k=n-1$ .**

6. Let  $G$  be a weighted graph with vertices  $A, B, C, D,$  and  $E,$  as well as several others. The shortest path from  $A$  to  $E,$  in terms of edges, is the path  $A,B,C,D,E.$  Suppose  $B, C,$  and  $D$  are articulation points of  $G.$  Does this imply that the path  $A,B,C,D,E$  is the shortest path from  $A$  to  $E?$  Prove your answer.

(10 points)

**No, give a counter-example.**

7. When finding an augmenting path for a maximum flow problem, it is sometimes necessary to move backwards along an edge to find an augmenting path. However, it is never necessary to go all the way back to the source. Prove it.

(10 points)

**Retaining such an edge would cause the flow out of the source to be incremented and decremented by the same amount, for no net change. The remainder of the path is, in itself, an augmenting path.**

8. Solve the following recurrence relations. Growth rate is sufficient.

$$T(n) = 7T\left(\frac{n}{10}\right) + n \lg n$$

$$\Theta(n \lg n)$$

$$T(n) = 27T\left(\frac{n}{3}\right) + n^3$$

$$\Theta(n^3 \lg n)$$

$$T(n) = 216T\left(\frac{n}{6}\right) + n^4$$

$$\Theta(n^4)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$

$$\Theta(n^2)$$

$$T(n) = 17T\left(\frac{n}{4}\right) + n^2$$

$$\Theta(n^{\log_4 17})$$

(10 points)

$$T(n) = 8T\left(\frac{n}{2}\right)$$

$$\Theta(n^3)$$

$$T(n) = 625T\left(\frac{n}{5}\right) + n^4$$

$$\Theta(n^4 \lg n)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$

$$\Theta(n^2)$$

$$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n^3}$$

$$\Theta(n^{3/2})$$

$$T(n) = 22T\left(\frac{n}{5}\right) + n^2$$

$$\Theta(n^2)$$

9. Differentiate the following functions.

$$f(n) = n^{\lg 7}$$

$$f'(n) = (\lg 7)n^{\lg 7 - 1}$$

$$f(n) = \binom{n}{3}$$

$$f'(n) = \frac{3n^2 - 6n + 2}{6}$$

$$f(n) = \sqrt[3]{7^{\lg n}}$$

$$f'(n) = \frac{(\ln 7)7^{\lg n}}{3n \ln 2 \sqrt[3]{7^{2 \lg n}}}$$

(10 points)

$$f(n) = \sqrt{e^{\lg n} + n^3}$$

$$f'(n) = \frac{1}{2\sqrt{e^{\lg n} + n^3}} \left( \frac{1}{n \ln 2} e^{\lg n} + 3n^2 \right)$$

$$f(n) = \lg^k n$$

$$f'(n) = \frac{\lg^{k-1} n}{n \ln 2}$$

$$f(n) = \sum_{i=0}^{\infty} n^i$$

$$f'(n) = \sum_{i=1}^{\infty} i n^{i-1}$$

10. Order the following functions by growth rate.

$$f(n) = n^{\lg 7}$$

$$f(n) = \binom{n}{3}$$

$$f(n) = \sqrt[3]{7^{\lg n}}$$

(10 points, all or nothing)

Fastest is first.

$$f(n) = \sum_{i=0}^{\infty} n^i \quad \Theta(\infty)$$

$$f(n) = n^{\lg 7} \quad \Theta(n^{2.81})$$

$$f(n) = \sqrt[3]{7^{\lg n}} \quad \Theta(n^{2.81/3})$$

$$f(n) = \sqrt{e^{\lg n} + n^3}$$

$$f(n) = \lg^k n$$

$$f(n) = \sum_{i=0}^{\infty} n^i$$

$$f(n) = \binom{n}{3} \quad \Theta(n^3)$$

$$f(n) = \sqrt{e^{\lg n} + n^3} \quad \Theta(n^{1.5})$$

$$f(n) = \lg^k n \quad \Theta(\lg^k n)$$