CS4321 Introduction to Algorithms

Final Exam

(100 possible points)
December 21, 2006

Name:

Problem	1	2	3	4	5	6	7	8	9	10	Total
Point											

Attention: Choose 3 out of Problems 7 to 10. If you work on all of them, the lowest grade will be dropped.

1. (10 points) The following questions are related to quicksort.
    a. (5 points) What is the worst case cost of quicksort? When does it happen?
    b. (5 points) If we use quicksort to sort array T[9] = {15, 21, 11, 9, 17, 7, 20, 1, 19}. Show the content of array T after first time pivot(), i.e., pivot(T, 0, 8), is called. What value does this call return?

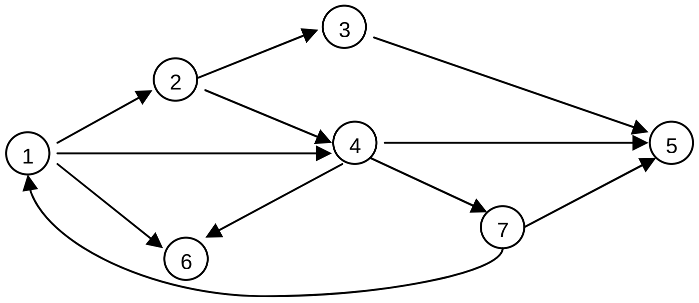
```
int pivot(T, i, j)
{
  // choose T[i] as the pivot
  p = T[i];
  l = i;   //  left cursor
  r = j+1;   // right cursor
  do {
    l++;
  } while (T[l]<=p and l <
r)

  do {
    r--;
  } while (T[r] > p);

  while (l<r) {
    swap(T[l], T[r]);
    do {
       l++;
    } while (T[l] <= p);
     do {
       r--;
     } while (T[r] >  p)
  }
  swap(T[i], T[r]);
  return r;
}
```

2. (14 points) In chained matrix multiplication algorithm, an array m[][] is used to track the optimal solution (minimum number of scalar multiplications) and an array p[][] is used to keep track of the split point for the optimal parenthesization. The table below shows the process of the algorithm for calculating the product of ABCD where A is 5×9, B is 9×2, C is 2×14 and D is 14×3. The value in cell (i, j) is m[i][j] and the value in the bracket is p[i][j].

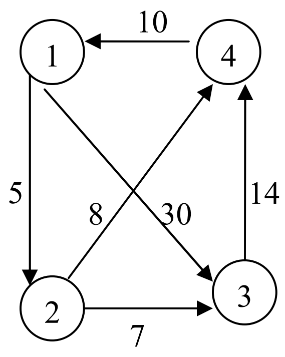
i　　j	1	2	3	4
1	0	90 (1)	230 (2)	
2		0	252 (2)	138 (2)
3			0	84 (3)
4				0

a. (4 points) Calculate m[1][4] and p[1][4] for cell (1, 4).
b. (4 points) Write a pseudo-code algorithm with p as your input to print the optimal parenthesization for which you can use Mi to denote the i-th matrix. An instance of your output for n=4 can be (M1(M2M3)M4).
c. (4 points) What's the cost of your algorithm? Justify your answer.
d. (2 points) What's the output of your algorithm for chain ABCD in this problem.

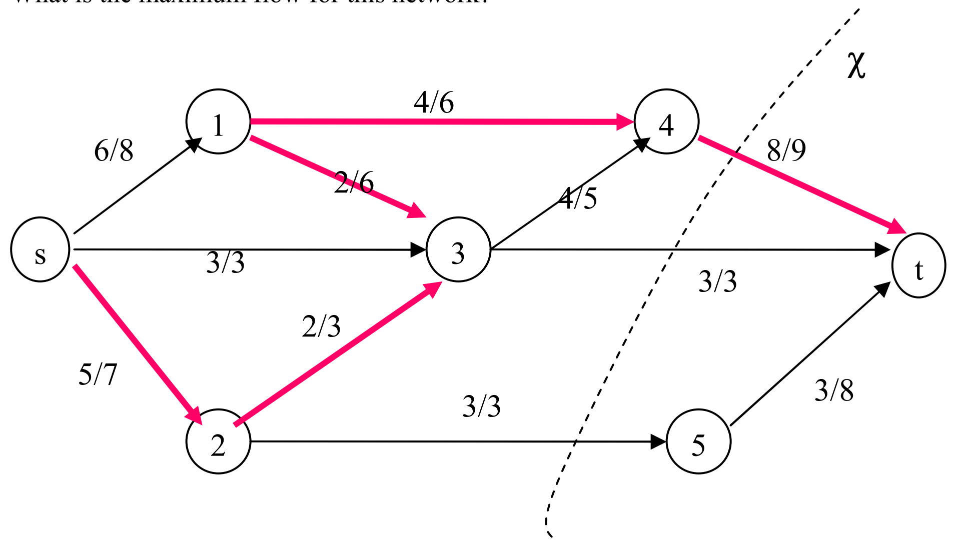
3. (15 points) We are running depth-first and breadth-first search on the graph below starting from node 1, and assuming that the searches visit the neighbors of a node in numerical order (smaller first).
   a. (4 points) What's the breadth-first ordering? Draw the breadth-first tree (or forest).
   b. (4 points) What's the depth-first ordering? Draw the depth-first search tree (or forest).
   c. (4 points) For your depth-first search, give the discovery time and finishing time of each vertex.
   d. (3 points) Does there exist a topological ordering for the graph? If yes, give an ordering. If no, justify.



4. (13 points) We are running one of the four algorithms on the graph below. (ignore the directions on the edges for Prim's and Kruskal's).
   - Prim's for the minimum spanning tree starting from node 1
   - Kruskal's for the minimum spanning tree
   - Dijkstra’s for single source shortest path where the source is node 1
   - Floyd’s all-pair shortest path
     a. (3 pts) Draw the minimum spanning tree generated by Prim's algorithm
     b. (2 pts) What is the second edge added by Kruskal's algorithm to the partial solution.
     c. (3 pts) Dijkstra’s algorithm uses an array D to record the distances of shortest special path. Given the content of D after the shortest path of node 2 is discovered.
     d. (2 pts) Give the adjacency matrix for the graph.
     e. (3 pts) What is D after the first iteration of Floyd’s algorithm, in other words, what is D_1?



5. (10 points) In the following flow network, each edge is annotated with its flow (on the left) and capacity (on the right).
    a. What is the capacity of cut χ as denoted as dashed line?
    b. What is the flow through cut χ?
    c. Is the path with the bolding edges an augmenting path? Why?
    d. What is the residual capacity of edge (1,3) with reprect to the path of the bolding edges.
    e. What is the maximum flow for this network?



6. (8 points) Given an array of four distinct elements, a, b, c, d, where the first two and the last two elements are already sorted, i.e., $a < b$ and $c < d$. Draw the decision tree to describe a comparison-based algorithm using least possible comparisons in worst case to find the first and the second **largest** elements. How many comparisons do you need in the worst case? Justify your decision tree uses the least number of comparisons in worst case. (Hint: think about mergesort)

7. (10 points) Professor Greedy drives an automobile from Newark to Reno along Interstate 80. His car's tank, when full, holds enough gas to travel *n* miles, and his map gives distances between gas stations on his route. The professor wishes to make as few gas stops as possible along the way. Give an efficient method by which Professor Midas determine at which gas stations he should stop, and prove that your strategy yields an optimal solution.

8. (10 points) In a large city, there are *k* police stations and *k* coffee shops. The human resources department wants to determine whether there are *k* disjoint paths from the police stations to the coffee shops (if not, perhaps it can be arranged). The paths must not contain any street or even any street corner in common. Show how to use network flows to solve this problem. Clearly describe how to model the situation with a flow network, including edge directions, capacities, source and sink, and explain how a maximum flow in the network answers the original question.

9. (10 points) You are given an array *A* with *n* values. You are told that the array *A* had previously been sorted, before it was rotated by some unknown number of positions. Provide an O(log n)-time divide-and-conquer algorithm that determines the maximum value in the array. You need to write a pseudo-code algorithm and show it is in O(log n). Also trace your algorithm for this array *A*, where *n* =17. [Here A was rotated 6 positions left or 11 positions right, but assume this is unknown.]

17	19	23	29	31	37	41	43	47	53	59	2	3	5	7	11	13

10. (10 points) Let X = {$x_1, \ldots ,x_n$} be a sequence of numbers. A subsequence of X is a sequence Y = {$x_{i1},\ldots,x_{ik}$} such that $1 \leq i1 < i2 < \ldots < ik \leq n$; that is, the elements of Y occur in X in the same order, but are not necessarily contiguous in X. A sequence is said to be increasing if each element is strictly larger than the previous. For example, a longest increasing subsequence of {7, 2, 3, 8, 5, 10, 14, 14, 7} is {2, 3, 8, 10, 14}.
   a. (4 points) Let C[i] denote the length of the longest increasing subsequence of X in which the last element is x_i. Thus, for the example above, C[9] = 4, because the longest increasing subsequence ending in x_9 (= 7) is {2, 3, 5, 7}. Give a dynamic programming recurrence for C[i].
   b. (4 points) Give an efficient dynamic programming algorithm to compute the length of the longest increasing subsequence of X. Please write your algorithm in pseudo-code and explain how it works.
   c. (2 points) What's the cost of your algorithm?