Name:_____ ID:_____

CSE 246 Analysis of Algorithms

Spring 2012 Midterm Exam 2

16.05.2012 Wednesday, Duration: 90 minutes

Q1	Q2	Q3	Q4	SUM
/25	/25	/25	/25	/100

Q-1. (25 pts) Consider a text with *n* zeros.
(a) (8 pts) How many character comparisons (in terms of *n*) will the Horspool's algorithm make in searching the pattern 010?

Shift Table:

0	1
2	1

The algorithm repeatedly makes 2 comparisons (1 doesn't match and 0 matches) and shifts by 2.
Total number of comparisons: n-2, for even values of n, and n-1, for odd values of n.

(b) (8 pts) How many character comparisons (in terms of *n*) will the brute-force string matching algorithm make in searching the pattern 010?

The algorithm repeatedly makes 2 comparison and shifts by 2.
Total number of comparisons: 2(n-2).

(c) (5 pts) What is the worst case input pattern of length 3 (3 bits) for the Horspool's algorithm? (To search in the text of *n* zeros)

100 (3 comparisons and shift by 1)

(d) (4 pts) What is the worst case input pattern of length 3 (3 bits) for the brute-force algorithm? (To search in the text of *n* zeros)

100 if the algorithm starts comparing from right.
001 if the algorithm starts comparing from left.
Both answers are accepted.

Name:_____ ID:_____

Q-2. (25 pts) Remember the knapsack problem: Given n items of known weights w_1, w_2, ..., w_n and values v_1, v_2, ... v_n and a knapsack of capacity W, the goal is to find the most valuable subset of the items that fit into the knapsack.
Now, consider a modified version of the knapsack problem: There are unlimited quantities of copies for each of the n item kinds given. In other words, you can pick from each item as many as you can (instead of only one).

(a) (10 pts) Design a dynamic programming algorithm for this modified version of the knapsack problem. (Hint: The problem is similar to the change making problem.)

$F(W) = \max_{j:W \geq w_j}\{F(W - w_j) + v_j\}$.
$F(W) = 0 \; if \; W < w_j, \;\; \forall j \in [1, 2, \ldots, n]$.

Using this recurrence, the algorithm can fill the one-row table in the same way as the change making problem.

(Remember that the recurrence for change making problem was $F(n)=\min\{F(n-d_j)+1\}$.)

(b) (10 pts) Apply your algorithm to the following instance:
$w_1 = 5$, $w_2 = 4$, $w_3 = 2$;
$v_1 = 10$, $v_2 = 4$, $v_3 = 3$;
$W=9$.

F(0)=0, F(1)=0,
F(2)=max{F(2-2)+3}=3,
F(3)=max{F(3-2)+3}=3,
F(4)=max{F(4-2)+3, F(4-4)+4}=6,
F(5)=max{F(5-2)+3, F(5-4)+4, F(5-5)+10}=10,
F(6)=max{F(6-2)+3, F(6-4)+4, F(6-5)+10}=10,
F(7)=max{F(7-2)+3, F(7-4)+4, F(7-5)+10}=13,
F(8)=max{F(8-2)+3, F(8-4)+4, F(8-5)+10}=13,
F(9)=max{F(9-2)+3, F(9-4)+4, F(9-5)+10}=16.

We can find selected items by backtracking.
When finding F(9), we use F(9-5)+10 as the maximum. So we use item 1.
When finding F(4) we use F(4-2)+3, so we use item 3.
Similarly, when finding F(2) we use F(2-2)+3, so we use item 3.
We use item 3 twice, and item 1 once.

(c) (5 pts) What is the time and space efficiency of your algorithm?

Time: O(W.n)

Space: O(W)

Name:_____ ID:_____

Q-3. (25 pts) Consider the modified version of the knapsack problem described in the previous question.

a. (10 pts) Design a greedy algorithm for solving the modified knapsack problem.

At each iteration, we select the item with highest v_i/w_i ratio, if the weight of the item doesn't exceed the remaining capacity. (If it exceeds, we select the item with highest ratio that fits to the knapsack). The algorithm terminates when the knapsack is full, or the remaining capacity is less than the weight of all items.

b. (7 pts) Apply your algorithm to the following instance (same instance as in question 2.b):

$w_1 = 5$, $w_2 = 4$, $w_3 = 2$; $v_1 = 10$, $v_2 = 4$, $v_3 = 3$; $W=9$

Does it give the optimal solution?

$v_1/w_1=2$; $v_2/w_2=1$; $v_3/w_3=1.5$,

First we pick item 1 (highest ratio). Remaining capacity=9-5=4

Second, we pick item 3 (since item 1 exceeds the capacity). Remaining capacity=4-2=2

Third, we pick item 3 again. Remaining capacity=2-2=0.

Value of the selected items = 10+3+3=16. It is optimal. (Same answer as 2.b.)

c. (8 pts) Does your algorithm always give the optimal solution? If yes, prove. If no, give a counter-example.

No. Counter example:

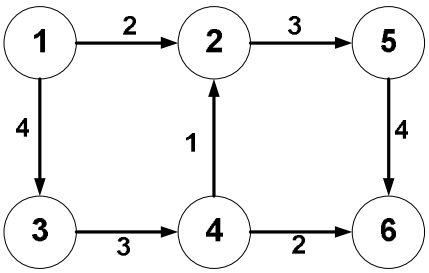
Reduce weight to 8 (W=8) and v_1 to 8.

The output of the algorithm will be: one item 1, one item 3 (total value = 8+3 = 11).

However, there is a better selection: four item 3 (total value = 4.3 = 12)

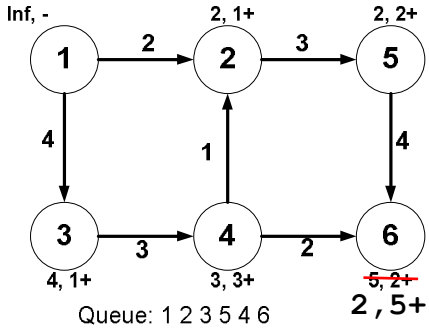
Name:_____ ID:_____

Q-4 (25 pts) Consider the following network:

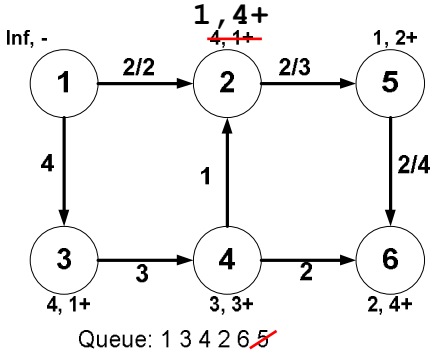


Node 1 is source and Node 6 is sink. Numbers on links are capacities.

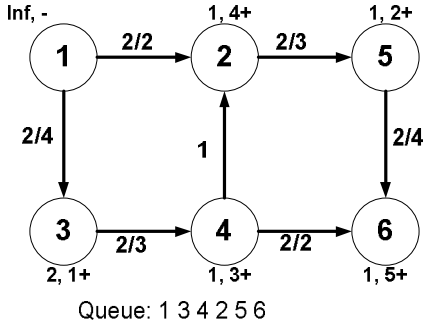
(a) (15 pts) Apply the shortest augmenting path algorithm to find the maximum flow?



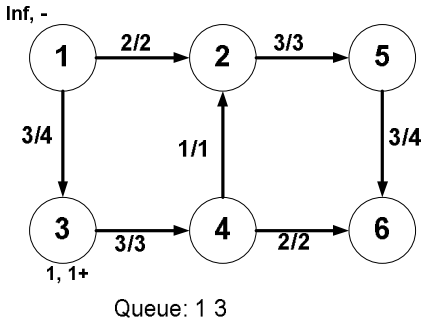
Augment the flow by 2 along the path
1 -> 2 -> 5 -> 6



Augment the flow by 2 along the path
1 -> 3 -> 4 -> 6



Augment the flow by 1 along the path
1 -> 3 -> 4 -> 2 -> 5 -> 6



No augmenting path (the sink is unlabeled). Maximum flow is 5.

(b) (10 pts) What is the minimum cut found by the shortest augmenting path algorithm? Are there other minimum cuts in the above network? If yes, how many? Show up all of them.

Minimum cut found by the algorithm is the edges from labeled vertices to the unlabeled vertices on the last iteration. Therefore, the algorithm finds the cut {(1,2),(3,4)}.

There are two othe minimum cuts: {(2,5),(4,6)}, {(1,2),(4,2),(4,6)}. Capacity of all these cuts are 5 which is equal to maximum flow.