

Question 1. [20 MARKS]

Short answer questions; no explanation required; all parts are worth 2 marks apiece; feel free to circle the correct answer.

- a. *True or false: If $f = \Theta(g)$, then $f^2 = \Theta(g^2)$*

True. If $0 < c < \frac{f(n)}{g(n)} < C$ for all $n \geq n_0$, then $0 < c^2 < \frac{f^2(n)}{g^2(n)} < C^2$ for all $n \geq n_0$.

- b. *True or false: If $\log f = \Theta(\log g)$, then $f = \Theta(g)$*

False. Counterexample: $f(n) = n$, $g(n) = n^2$.

- c. *True or false: $\binom{2n}{n} = O(2^n)$*

False.

$$\binom{2n}{n} = \frac{2n}{n} \cdot \frac{2n-1}{n-1} \cdots \frac{n+1}{1} \geq 2^{n-1} \cdot (n+1)$$

Therefore $\frac{\binom{2n}{n}}{2^n} \geq \frac{n+1}{2}$ is unbounded.

- d. *True or false: $2^{n+1} = O(2^n)$*

True. $\frac{2^{n+1}}{2^n} = 2$, i.e. it's bounded.

- e. *True or false: Let G be a weighted graph on n vertices. Then G has not more than n minimum spanning trees.*

False. A complete graph on 4 vertices has far more than 4 spanning trees (in fact, 16). If we assign the same cost to all 6 edges, then each of these trees is an MST. $A[1] = 0$

- f. *True or false: We have n tasks of unit length, each with its deadline. If the latest deadline is greater or equal to n , then all jobs can be scheduled on a single processor so that each one is completed by its deadline.*

False. Counterexample: $d_1 = d_2 = 1$ prevents simultaneous scheduling of the tasks ## 1 and 2.

- g. *True or false: No matter how it is parenthesized, the evaluation of the product of n matrices $A_1 \cdot A_2 \cdots A_n$ involves $n - 1$ matrix multiplications.*

True. Each multiplication reduces the number of terms yet-to-be-multiplied by 1.

- h. *True or false: Assume that in a flow network N each of the vertices s (source) and t (sink or target) is known to be an endpoint of one edge only, e_1 for s , e_2 for t . Then for any flow f in N , $f(e_1) = f(e_2)$.*

True. $f(e_1) = f(e_2) = |f|$ (consider the cuts through e_1 and through e_2).

- i. *Multiple choice: We are given an array of integers $A[1], \dots, A[n]$. The problem: fill out the array $S[i, j]$ where $S[i, j] = 0$ if $i > j$, and otherwise $S[i, j] = \sum_{k=i}^j A_k$. The running time for the most time-efficient algorithm for this problem is...*

$$\Theta(n)? \quad \Theta(n^2)? \quad \Theta(n^3)?$$

Answer: $\Theta(n^2)$. We merely need to add $A[j]$ to $S[i, j - 1]$ to get $S[i, j]$ (or subtract $S[1, i - 1]$ from $S[1, j]$), but we have to do at least $1/2 \cdot n^2$ ops just to write the correct values to the array.

- j. *Multiple choice: In the job scheduling with deadlines, durations, and profits problem given in class, given jobs J_i for $i = 1, \dots, n$ we used dynamic programming to construct the table A where $A[i, t]$ was the maximum profit obtainable from scheduling jobs J_1, \dots, J_i within time t . In order to extract an actual optimal schedule from this table (i.e. the start times of jobs J_1, \dots, J_n or -1 if the job is not scheduled) we require time...*

$$\Theta(n)? \quad \Theta(n^2)? \quad \Theta(n^3)?$$

Answer: $\Theta(n)$. The procedure for scheduling is called n times – once for each job. At each recall it performs a bounded number of operations, in addition to recalling itself. (See CSC364 notes.) Alternatively, there is a loop of length n (one iteration for every job), and the number of ops per iteration is bounded. (See Handouts – Lecture Notes – Week 5.)

Question 2. [15 MARKS]

In the *art gallery guarding* problem we are given a line L that represents a long hallway in an art gallery. We are also given a set $X = \{x_1, x_2, \dots, x_n\}$ of real numbers in no particular order that specify the positions of paintings in this hallway. Suppose that a single guard can protect all the paintings within distance at most 1 of his or her position (on both sides).

- a. *Design a greedy algorithm for finding a placement of guards that uses the minimum number of guards to make sure that all the paintings with positions in X are safe (let your solution be the vector $Y = \{y_1, \dots, y_k\}$ of guards' positions, where k is the minimum required).*

Sort X in increasing order.

for $i = 1$ to ∞ do

 if all paintings are safe then stop

 else place guard i at the position $x_j + 1$ where x_j is the leftmost unguarded painting

- b. *Prove that your algorithm is correct (you do not need to use the greedy template here, but feel free to do so if you think it applies).*

Let $z_i = y_i - 1$ for all $1 \leq i \leq k$. Then by virtue of the algorithm, there are paintings at all points z_i , $1 \leq i \leq k$, and the distance between any two of them is greater than 2. Therefore no guard can keep in sight two of the paintings z_i , $1 \leq i \leq k$ simultaneously. Therefore, there is no way to settle for less than k guards.

(The greedy template can be used as well.)

- c. *Give asymptotic bound for running time, assuming that any arithmetic operation, comparison, assignment, indexing into an array can be performed in time bounded by a constant. Provide some justification.*

Answer: $O(n \log n)$.

This is what the initial sorting of the points requires; after that we only need $O(n)$ for the loop that sets the guards in place.

Question 3. [15 MARKS]

There are n stations numbered 1 to n on a one-way railroad. The fares $c(i, j)$ are known for rides between any two stations i and j (for $i < j$; there is no service in the opposite direction). Your job is to find the minimum total cost of the journey between the first and the last station (transfers on the way may reduce this cost).

- a. Define an array indexed by the parameters that define subproblems, to store the minimum cost for each subproblem (make sure that one of the “sub”problems is actually equal to the whole problem).

$A[i, j]$, for $1 \leq i \leq j \leq n$, is the minimum cost of getting from station i to station j with transfers allowed. $A[1, n]$ then becomes the answer to the original problem.

- b. Based on the recursive structure of the problem, describe a recurrence relation satisfied by the array values from (a) (including degenerate or base cases).

$A[i, i] = 0$ for all $1 \leq i \leq n$,

$A[i, j] = \min\{c(i, j), \min_{i < k < j}\{A[i, k] + A[k, j]\}\}$ for all $1 \leq i < j \leq n$.

- c. Give an algorithm to compute the values in the array in a bottom-up fashion.

```

for  $i = 1$  to  $n$  do
     $A[i, i] \leftarrow 0$ 
for  $m = 1$  to  $n - 1$  do
    for  $i = 1$  to  $n - m$  do
         $j \leftarrow i + m$ 
         $A[i, j] \leftarrow c(i, j)$ 
        for  $k = i + 1$  to  $j - 1$  do
            if  $A[i, k] + A[k, j] < A[i, j]$  then
                 $A[i, j] \leftarrow A[i, k] + A[k, j]$ 
            end if
        end for
    end for
end for
return  $A[1, n]$ 

```

THE RUNNING TIME OF THE ALGORITHM ABOVE IS $O(n^3)$. SEVERAL STUDENTS CAME UP WITH THE SOLUTION BELOW THAT IS SIMPLER AND MORE EFFICIENT. ITS RUNNING TIME IS $O(n^2)$.

WE THANK KEN YUE HON CHAN FOR BRINGING THIS ALGORITHM TO OUR ATTENTION.

- a. Define an array indexed by the parameters that define subproblems, to store the minimum cost for each subproblem (make sure that one of the “sub”problems is actually equal to the whole problem).

$A[i]$ ($1 \leq i \leq n$) – the minimum cost of getting from station 1 to station i (transfers allowed). $A[n]$ is the answer to the original problem.

- b. Based on the recursive structure of the problem, describe a recurrence relation satisfied by the array values from (a) (including degenerate or base cases).

$A[1] = 0$,

$A[i] = \min_{1 \leq k < i}\{A[k] + c(k, i)\}$ for all $1 < i \leq n$.

c. Give an algorithm to compute the values in the array in a bottom-up fashion.

```
A[1] ← 0
for i = 2 to n do
    A[i] ← min1 ≤ k < i{A[k] + c(k, i)}
end for
return A[n]
```

Total Marks = 50