100 marks **CS 301 Midsemester Test** **9:30-11:30, 21/9/5**

Problem 1[10 marks] Solve the recurrence: $S(n) = 1 + 2S(n/4)$, $S(1) = 1$. It is enough if you give an answer to within constant factors, say as $S(n) = O(\ldots)$.

$S(n) = 1+2S(n/4) = 1+2+4S(n/16) = 1+2+4+\ldots+2^{k-1}+2^kS(n/4^k) = 2\cdot 2^k-1 = 2\sqrt{n}-1$

Master theorem cannot be used – as mentioned during examination.

Proof by guessing + induction is a bit tricky. The simple hypothesis $S(n) \leq c\sqrt{n}$ does not work. You need $S(n) \leq c\sqrt{n} - a$ and then show that $a = 1$ suffices. A proper proof by induction gets full 10 marks.

Drawing out the recursion tree and adding up from it is sufficient as well.

But if the algebra is wrong you could loose at least 5 marks unless the mistake is trivial. e.g. getting a solution of $1 + 2\log_4 n + \sqrt{n}$ as some have will get 5 marks.

Problem 2[25 marks] Suppose there are n projects on which you can work. There are a total of H hours you can spend on these projects. The returns you get on each project are given to you as an $n \times H$ matrix R, where $R[i, j]$ denotes the return (in rupees) you get if you spend j hours on project i. The returns need not be linear, i.e. doubling the time spent on a project may not double the return.

Give an algorithm that takes as input integers n, H and the matrix R and determines how many hours to allocate to each project so that the total return (the sum of the returns on each project) is maximized.

$$G[i, H] = \max_{r \in [0,H]} \{R[i, r] + G[i + 1, H - r]\}$$

G[i,H] is the maximum return for projects i through n using H hours. G[n,j]=R[n,j] for all j. Thus every entry takes time O(H) to fill. Total of nH entries, so time $O(nH^2)$.

Problem 3[25 marks] An RNA molecule is represented as a string $S[1..n]$ over the alphabet $\mathcal{A} = \{A, C, G, U\}$ (respectively denoting the *bases* Adenine, Cytosine, Guanine, and Uracil). A *folding* of an RNA molecule is a sequence $\{(L_1, R_1), \ldots, (L_m, R_m)\}$, with the pair (L_i, R_i) denoting a bond between the L_ith and the R_i th base in the molecule, for $1 \leq L_i < R_i \leq n$. Each bond (L_i, R_i) is associated with an energy drop $E(S[L_i], S[R_i])$, where $E : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is a known function, say available as a 4×4 table. The total energy drop of the folding is simply the sum of the energy drops of the bonds in it. Foldings must satisfy certain restrictions: (i) Each base of the molecule can appear in at most 1 bond, i.e. the $2m$ numbers L_i, R_i must be distinct, and (ii) Bonds cannot cross, i.e. for any i, j, the order $L_i < L_j < R_i < R_j$ is PROHIBITED (but others, e.g. $L_i < R_i < L_j < R_j$ or $L_i < L_j < R_j < R_i$ are allowed). Develop an algorithm which given S finds the folding having the maximum energy drop.

base 1 can form a bond with any other base from 3 onwards, or not form a bond at all.

S_i : space in which 1 froms a bond with i. i=0 means that i does not bond at all.

Best configuration in S_i : Best for 2..i and i+1..n.

Best S_0 : Best for 2..n

$D[i,j]$ = max drop for sequence i..j

$$D[i,j] = \max(D[i+1,j], \max_{k=i+2..j}\{E[i,k] + D[i+1,k-1] + D[k+1,j]\})$$

$$D[i,i+1] = 0 \quad D[i,i] = 0$$

There will be $O(n^2)$ entries. Subsequently filling the entries will take time O(n) for each entry, so total time is $O(n^3)$.

Basic recurrence: 15 marks. Base case: 5 marks. Time estimate: 5 marks.

Problem 4[30 marks] You have a machine which can manufacture 1 unit of a certain commodity every day. You are given a sequence of numbers $D = \{d_1, \ldots, d_n\}$ in which d_i represents the number of units that are to be delivered on the evening of the ith day. If more than 1 unit is to be delivered on some day, your manufacture must begin earlier, obviously. However, any manufactured units that remain undelivered at the end of any day must be held in a warehouse. The cost of holding 1 unit in the warehouse for 1 night is H. In addition, if the machine is not working on day i but needs to work on day $i+1$, then there is a startup cost S that is also incurred. You are to devise an algorithm that takes D, S, H as input and plans the production so that the total cost: holding cost and startup cost is minimized. Initially, the machine is off, and there are no previously manufactured units. At the end of the n days, all the units manufactured must have been delivered. If the given demand cannot be met, your algorithm should indicate as much.

As an example, suppose $D = \{0, 0, 2, 0, 0, 0, 3\}$, $H = 1$ and $S = 5$. One production plan would be to run the machine on days 2,3 and 5,6,7. This has holding costs of 1,1,2 on days 2,5,6 respectively and startups on day 2 and day 5. The total cost is thus 1+1+2+5+5=14. A better plan is to run the machine on days 2,3,4,5,6. This will have an extra holding cost but one less startup, for a total cost of 12.

We will have a table C[1..n+1,-n..0..n,0..1].

$C[i,N,m]$: cost of fullfilling the demand with N items in stock and the status m of the machine as given (0/off,1/on) starting from day i.

Base cases:

$C[n+1,0,m] = 0$: Nothing left over on day $n+1$.

$C[n+1,N \neq 0,m] = \infty$: Nothing must be left over on day $n+1$.

$C[i,N,m] = \infty$ if $N < 0$ for all i, m. Inventory cannot become negative on any day.

Filling an entry:

$$C[i,N,m] = \min(H(N+1-d_i)+C[i+1,N+1-d_i,1]+S(1-m), H(N-d_i)+C[i+1,N-d_i,0]+0)$$

The only decision to be made on day i is whether to produce that day, or not. The first term in the $\min(\ldots)$ represents the cost if an item is produced that day, and the second term, if not. The three addends in each term, respectively, are the holding cost, the cost incurred subsequetly, and the startup cost.

Time to fill any single entry in the table = O(1). There are a total of $O(n^2)$ entries. Hence the time = $O(n^2)$.

Correct basic recurrence (filling an entry) : 15 marks.

Base cases: 10 marks.

Time estimate: 5 marks.

Problem 5 Consider a 2^n letter alphabet. Suppose the frequency of the ith letter is $2^n + i$. (a)[7 marks] For the case $n = 3$, show the Huffman code generated. (b)[3 marks] Speculate on what happens for general n, giving your reasons in AT MOST 5 SENTENCES.

(a) The frequencies are $9, 10, \ldots, 16$. We will get a fixed length 3 bit code eventually.

5 marks if it clear that the method is understood. 0 if it is not clear. As much as possible no partial credit. Full marks if the final answer is guessed correctly after doing some of the calculation only.

2 marks if the final answer is correct. No partial credit.

(b) We will get a fixed length n bit code. After mergeing the two least frequency letters, the new letter formed will have the highest frequency, and this will continue till the end. So every letter will be merged equal number of times. Hence a fixed length code will result.

1 mark for stating fixed length code, 2 marks for the rationale. only 1 mark if you only observe that the frequencies are nearly the same/less than a factor of two different.