

1. Recurrence [8%]

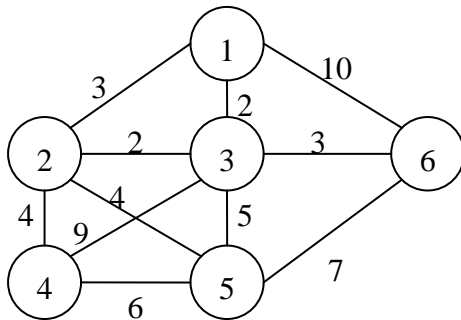
Please find the asymptotic upper and lower bounds for  $T(n)$  in each of the following recurrences. Assume that  $T(n)$  is constant for sufficiently small  $n$ . Make your bounds as tight as possible, and justify your answers.

- I.  $T(n) = 5T(n/2) + n^2$
- II.  $T(n) = 3T(n/2) + n \log n$
- III.  $T(n) = 9T(n/3) + n^2 \log n$
- IV.  $T(n) = 16T(n/2) + n^5$

Ans:

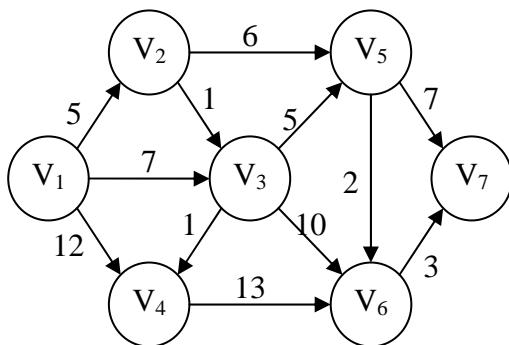
- I.  $T(n) = O(n^{\log 5})$
- II.  $T(n) = O(n^{\log 3})$
- III.  $T(n) = O(n^2(\log n)^2)$
- IV.  $T(n) = O(n^5)$

2. Please find the minimum spanning tree of the following graph, step by step.



- I. Use Kruskal's algorithm. [5%]
- II. Use Prim's algorithm (initial vertex is 1) [5%]

3. Use the Dijkstra's algorithm to find shortest paths from  $V_1$  to all other nodes.



[6%]

4. Please answer the following questions. [16%]

- I. “Divide-and-Conquer” and “Dynamic Programming” strategies both use the sub-problem’s solution to get global solution. What is the difference between these two strategies?
- II. What is the difference between “Divide-and-Conquer” and “Prune-and-Search”?
- III. If a problem has the principle of optimality property, we can use dynamic programming strategy to solve it. What is the principle of optimality?
- IV. Please explain the two important mechanisms of branch-and-bound strategy.

Ans:

- I. The main difference is that sub-problems are more or less independent in divide and conquer, where as the overlap of sub-problems occur in dynamic programming.
- II. “Divide and Conquer” divides the problem into sub-problem, but “Prune-and-Search” prunes the unfeasible solution.
- III. Suppose that in solving a problem, we have to make a sequence of decisions  $D_1, D_2, \dots, D_n$ . If this sequence is optimal, then the last  $k$  decisions,  $1 < k < n$  must be optimal.
- IV. Branch: Generate branching.  
Bound: Generate a bound so that many branching can be terminated.

5. Greedy -- Optimal expected program storage [15%]

Assume  $n$  programs of lengths  $(L_1, L_2, \dots, L_n)$  are to be stored on a tape. Program  $i$  is to be retrieved with probability  $P_i$ . If the programs are stored in the order  $I = (I_1, I_2, \dots, I_n)$ , the expected retrieval time (ERT)  $= \sum_{i=1}^n P_{I_i} \sum_{j=1}^i L_{I_j}$ . We want to find an optimal order  $I$  which minimizes the ERT. Please use greedy strategy to solve this problem, and prove its correctness and time complexity.

Ans: We place the programs according to the value  $R_i = L_i/P_i$ , sorting it into a non-decreasing sequence. It must be optimal solution, and it’s time complexity is  $O(n \log n)$ .

Let  $I = i_1, i_2, \dots, i_n$  be any permutation of the index set  $\{1, 2, \dots, n\}$ .

$$ERT(I) = \sum_{i=1}^n P_{I_i} \sum_{j=1}^i L_{I_j}$$

If there exist  $a$ , and  $b$  such that  $a < b$  and  $b$  is the smallest index such that  $I_a = A$ ,  $I_b = B$ ,  $R_A > R_B$ , then interchanging  $I_a$  and  $I_b$  results in a permutation  $I'$ .

We let  $P_x = \sum_{i=a+1}^{b-1} P_{I_i}$ ,  $L_x = \sum_{i=a+1}^{b-1} L_{I_i}$

$$\begin{aligned} \text{ERT}(I) - \text{ERT}(I') &= (L_A - L_B)(P_x + P_A + P_B) + (P_B - P_A)L_x + P_B L_B - P_A L_A \\ &= (P_B L_A - P_A L_B) + P_x(L_A - L_B) + (P_B - P_A)L_x \end{aligned}$$

Since  $R_A > R_B$ ,  $P_B L_A - P_A L_B > 0$ .

Because  $L_x/P_x > R_B$ ,  $L_x > P_x L_B/P_B$

$$\begin{aligned} P_x(L_A - L_B) + (P_B - P_A)L_x &> P_x(L_A - L_B) + (P_B - P_A)P_x L_B/P_B \\ &> P_x(L_A - P_A L_B/P_B) \\ &> 0 \end{aligned}$$

We get  $\text{ERT}(I) - \text{ERT}(I') > 0$ .

So, no permutation that is not in non-decreasing order of the  $R_i$  can have minimum ERT. And all permutations in non-decreasing order have the same ERT. So this order minimizes the ERT.

#### 6. Divide-and-Conquer – Integer Multiplication [15%]

Multiplying two  $n$  bit numbers  $u$  and  $v$  straightforwardly requires  $O(n^2)$  steps. Please use divide-and-conquer strategy to solve this problem, and analyze the time complexity.

Ans: Let  $u = (a \cdot 2^{n/2} + b)$ ,  $v = (c \cdot 2^{n/2} + d)$

$$uv = (a \cdot 2^{n/2} + b) * (c \cdot 2^{n/2} + d) = ac \cdot 2^n + (ad + bc) \cdot 2^{n/2} + bd.$$

$ad + bc$  is computed as  $(a+b)(c+d) - ac - bd$ .

So we can get the recurrence relation  $T(n) = 3T(n/2) + O(n)$

$$T(n) = O(n^{\log 3})$$

#### 7. Prune-and-Search – Constrained Linear Programming with Two Variables [15%]

The simplified two-variable linear programming problem is defined as follows:

Minimize  $y$

Subject to  $y \geq a_i x + b_i, i = 1, 2, \dots, n.$

Please use prune-and-search strategy to solve this problem, and analyze time complexity.

Ans: See textbook.

#### 8. Dynamic programming – Longest Common Subsequence [15%]

A subsequence of string  $S$  is obtained by deleting 0 or more (not necessarily consecutive) symbols from  $S$ . A common subsequence between  $A$  and  $B$  is defined to be subsequence of both strings. The longest common subsequence

problem is to find a longest common subsequence between two strings. Given two strings  $A = a_1a_2\dots a_n$  and  $B = b_1b_2\dots b_m$ , please use dynamic programming strategy to solve the LCS problem, and analyze the time and space complexity.

Ans: See textbook.