Second Midterm Exam of Design and Analysis of Algorithms

May 11, 2022

1. (10%) The quick sort is a divide and conquer algorithm and its worst-case time complexity is known to be $O(n^2)$, where n is the number of the input elements. Please modify the algorithm of the quick sort so that its worst-case time complexity becomes $O(n \log n)$.

2. (15%) Please find the *tight* asymptotic upper bounds of the following recurrences in big-O notation and justify your answers. Note that you can use the master theorem to justify your answers.

   (a) (5%) $T(n) = 2T\left(\frac{n}{3}\right) + 6n$

   (b) (5%) $T(n) = 2T\left(\frac{n}{2}\right) + 6n$

   (c) (5%) $T(n) = T\left(\frac{3n}{4}\right) + T\left(\frac{n}{4}\right) + n$

3. (10%) The *maximum subarray sum problem* is to find a contiguous subarray with the largest sum within a given a one-dimensional array $A = [a_1, a_2, \ldots, a_n]$ of n numbers. For example, if $A = [1, -2, 5, -3, 4, 8, -6, 7]$, then the contiguous subarray with the largest sum is $[5, -3, 4, 8]$ and its sum is 14. Please design a divide and conquer algorithm whose time complexity is better than $O(n^2)$ to solve the maximum subarray sum problem (5%) and analyze its time complexity (5%).

4. (10%) Let u and v be two n bit binary numbers, where n is assumed to be a power of 2 for simplicity. Clearly, multiplying u and v straightforwardly requires $O(n^2)$ steps. In other words, the time complexity of the straightforward method if $O(n^2)$. Please design a divide and conquer algorithm whose time complexity is better than $O(n^2)$ for multiplying u and v (5%) and also analyze its time complexity (5%).

5. (10%) Determine whether the following statements are correct and also justify your answers.

   (a) (5%) When we design a divide and conquer algorithm for solving a problem, dividing the problem into more than two equal-size sub-problems is always better than dividing it into two equal-size sub-problems from the viewpoint of their big-O time complexities.

   (b) (5%) The solution obtained by the branch and bound algorithm introduced in the class for solving the 0/1 knapsack problem actually is not optimal, although its worst-case time complexity is polynomial.

6. (10%) Please use the hill-climbing method (5%) and the best-first search method (5%) respectively to solve the 8-puzzle problem with the following initial state (left) and final goal state (right). You need to define your evaluation functions for these two methods and also draw their searching trees.

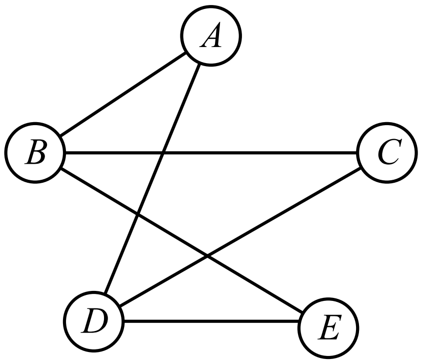
3	8	2
6		1
4	5	7

Initial state

3	2	1
4		8
5	6	7

Goal state

7. (10%) A graph is 2-*colorable* if and only if all its vertices can be colored using two colors such that no two vertices connected by an edge have the same color. Consider the following graph $G = (V, E)$. Is this graph 2-colorable? Please answer this question by some kind of tree searching algorithm (6%). Note that you should explain the reason why your searching algorithm is correct. Please also analyze the time complexity of your searching algorithm (4%).



8. (5%) Explain the reason why the A^* algorithm can be considered as a special kind of branch and bound algorithm.

9. (10%) Solve the traveling salesperson problem characterized by the following distance matrix by the branch and bond algorithm we introduced in the class. Please write down the solution of this TSP instance (3%) and its final search tree of the branch and bound algorithm (7%).

i \ *j*	**1**	**2**	**3**	**4**	**5**
1	∞	6	62	35	13
2	57	∞	43	20	7
3	40	43	∞	9	22
4	6	50	43	∞	8
5	42	27	11	36	∞

10. (10%) Given the list of code words $\{000000, 100110, 001011, 101101, 010101, 110011, 011110, 111000\}$ and a received vector $(r_1, r_2, \dots, r_6) = (-2, 1, 2, -2, -3, -1)$, please use the A^* algorithm to find a path from the root of the code tree to a goal node such that the cost of the path is minimum among all paths from the root to a goal node. Note that in your answer, you need to also draw your searching tree for the given instance.