Final Exam of Design and Analysis of Algorithms

June 24, 2015

1. (10%) Consider the 8-puzzle problem with the following initial state (left) and goal state (right). Solve this 8-puzzle instance by using the hill-climbing method (5%) and the best-first search method (5%). You need to define your evaluation functions for these two methods and also draw their searching trees.

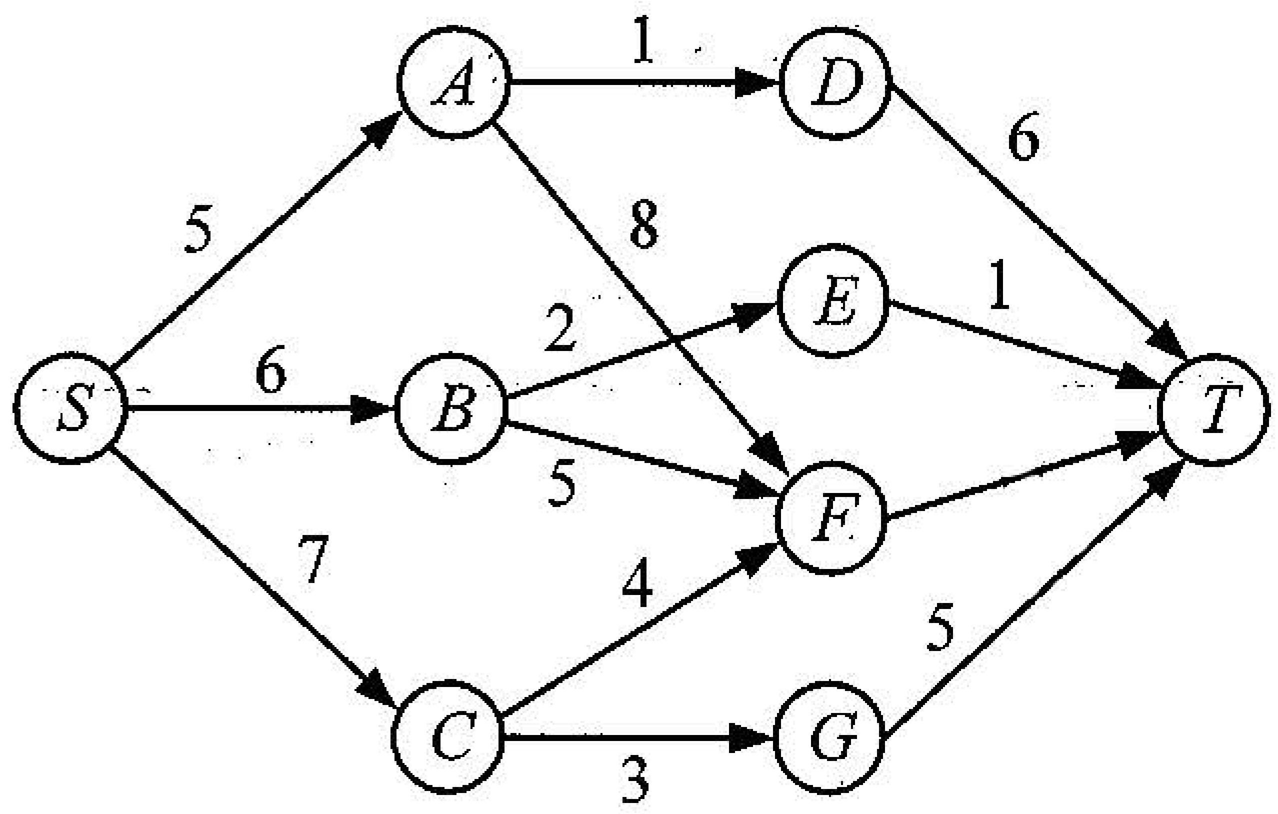
2	3	
8	1	4
7	5	6

Initial state

1	2	3
8		4
7	6	5

Goal state

2. (5%) Find the shortest path from S to T by using the A^* algorithm. Please also draw your searching tree for the given instance.



3. (5%) Consider the following 0/1 knapsack instance:

i	1	2	3	4
P_i	10	10	12	18
W_i	2	4	6	9
M	15			

Solve this 0/1 knapsack instance by the branch-and-bound strategy. Please also draw your searching tree for the given instance.

4. (10%) What is the so-called prune-and-search strategy?

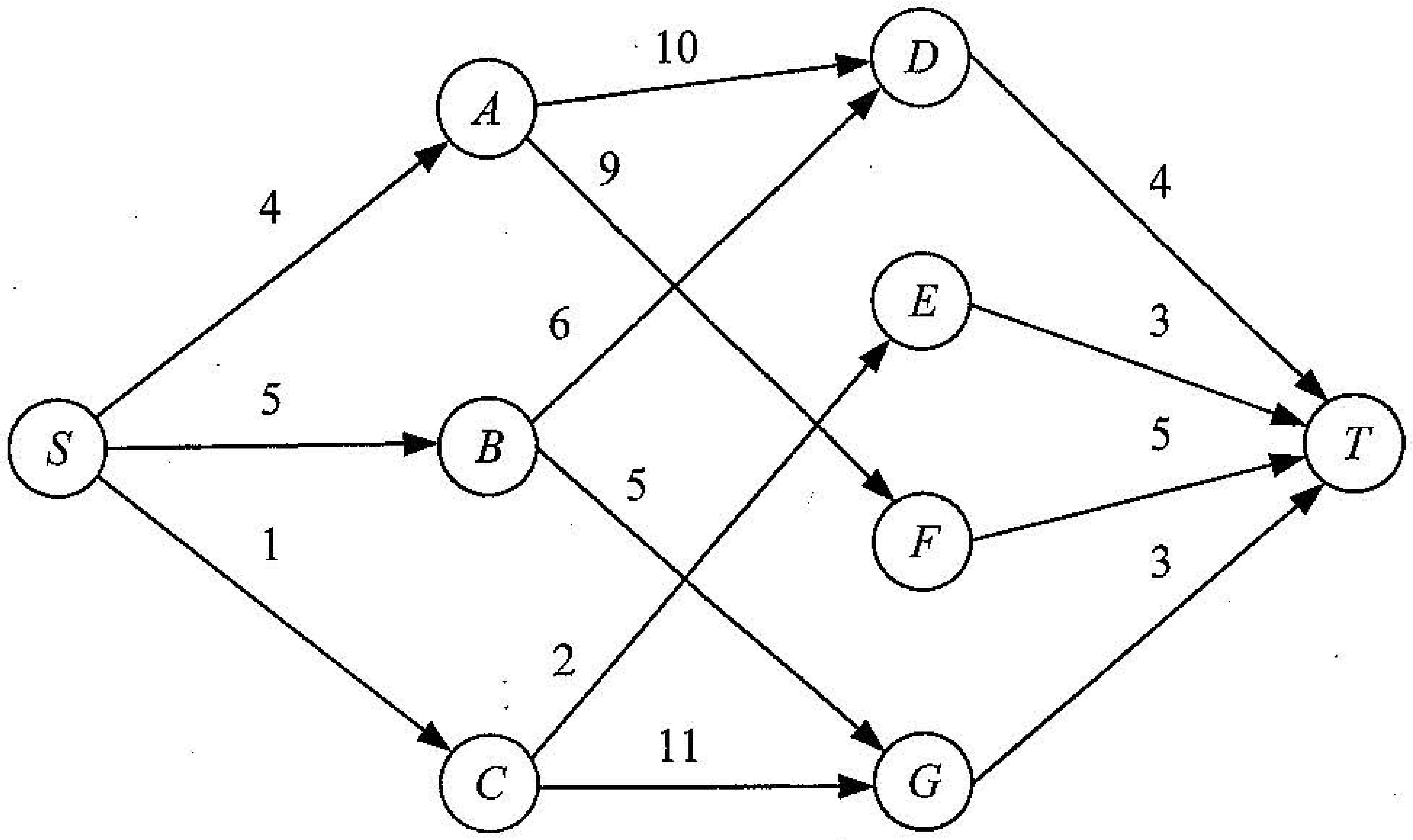
5. (5%) Let $T(n) = T\left(\frac{3n}{4}\right) + T\left(\frac{n}{7}\right) + 2n^2$. Please find the asymptotic upper bounds of $T(n)$ in big-O notation.

6. (5%) Input are a sequence S of n distinct values, not necessarily in sorted ordered, and two integers m_1 and m_2, where $1 \leq m_1, m_2 \leq n$. For any value x in S, we define the rank of x in S to be $|\{k \in S: k \leq x\}|$. Please design an algorithm to output all the values of S whose ranks fall in the interval $[m_1, m_2]$ in $O(n)$ time.

7. (5%) What is the so-called principle of optimality? (2%) Find a problem for which the principle of optimality does not hold and also explain the principle does not hold. (3%)

8. (5%) Describe the differences (3%) and similarities (2%) between divide-and-conquer and dynamic programming methods.

9. (5%) Consider the following graph and find the shortest path from S to T by the dynamic programming approach.



10. (5%) Suppose that X, Y and Z are three given strings, where $|X| = n$, $|Y| = m$ and $|Z| = n + m$. Then Z is said to be a *shuffle* of X and Y if and only if Z can be formed by interleaving the characters from X and Y in a way that maintains the left-to-right ordering of the characters from each string. For example, $abcdef$ and $acebdf$ are shuffles of ace and bdf, but $aecbdf$ is not. Please use the dynamic programming approach to design a polynomial-time algorithm to determine whether Z is a shuffle of X and Y. Please also analyze the time complexity of your algorithm.

11. (5%) Please answer the following questions.
    (a) (2%) What is NP-complete?
    (b) (3%) Please draw a diagram to show the commonly believed relationships among P, NP, NP-complete and NP-hard.

12. (5%) Prove that the partition problem can reduce to the bin packing decision problem. These two problems are defined as follow: Given a set of n positive integers $A = \{a_1, a_2, \dots, a_n\}$, the partition problem is to determine whether there is a partition $A = \{A_1, A_2\}$ such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$. Given a set of n items, each of size c_i which is a positive integer, and two positive integers B and C which are the number of bins and the bin capacity respectively, the bin packing decision problem is to determine whether we can assign the n items into B bins such that the sum of c_i's over all items assigned to each bin does not exceed C.

13. (5%) Prove that the Halting problem is NP-hard.

14. (5%) Prove that the satisfiability problem with each clause containing at most 3 literals, denoted by $\leq$3SAT, is NP-complete.

15. (20%) Determine whether the following statements are correct or not. **If not, please explain your reason (<u>no reason, no point</u>).**

(a) (2%) In the worst case, the time complexity of the branch-and-bound algorithm for solving the traveling salesperson problem is polynomial.

(b) (2%) In an ordinary A^* algorithm, we can terminate the algorithm when a goal node is produced.

(c) (2%) If $T(n) = T\left(\frac{3n}{4}\right) + n^2$, then $T(n) = O(n)$.

(d) (2%) A dynamic programming algorithm saves its computational time by eliminating solutions and avoiding computing the same subproblems repeatedly.

(e) (2%) The 0/1 knapsack problem can be solved by a dynamic programming algorithm in polynomial time

(f) (2%) The Cook's theorem states that if the SAT problem is in NP, then P = NP.

(g) (2%) For an NP-complete problem P, we need to take exponential time to solve this problem P for all kinds of inputs.

(h) (2%) An optimization problem is NP-hard if its corresponding decision problem is NP-complete.

(i) (2%) If problems P_1 and P_2 are known to be NP-hard, then we can conclude that $P_1 \propto P_2$ and $P_2 \propto P_1$.

(j) (2%) Both the 2-SAT and 3-SAT problems are NP-complete.