CS300 Introduction to Algorithm (Sample Problems for Final)

1. (a) Define the minimum cost spanning tree(MST).

   (b) Describe the Kruskal's Algorithm and give the runtime of that.

   (c) Prove the correctness of Kruskal's Algorithm.

   (d) What is the strategy for the Kruskal's algorithm.

Ref: CS300, 2017 Spring

Edited by 20170504 Juan Lee

2. (a) What is greedy strategy?

   (b) Give some examples where greedy doesn’t work.

3. (a) State the UNION-FIND Problem.

   (b) Give the way to implement UNION-FIND and way of improvement.

   (c) Prove that the way is improved way.

   (d) Give the runtime of improved algorithm.

   (e) Suggest improvement of UNION-FIND using “path-compression”

4. Briefly define alphabet, character, strings(or words), length, empty word, prefix, suffix, substring.

5. State the brute forcing algorithm for string matching and the runtime of that.

6. (a) Describe Rabin-Karp Algorithm.

   (b) Give the runtime of Rabin-Karp Algorithm.

   (c) What can be problem for the Rabin-Karp Algorithm?

   (d) Give the remedy to solve the problem and the problem of the remedy and solve again it.

   (e) Give the runtime of improved Rabin-Karp Algorithm.

7. (a) Describe Boyer-Moore Algorithm.

   (b) Give the runtime of Boyer-Moore Algorithm.

   (c) Suggest “Suffix Heuristic” for Boyer-Moore Algorithm. Explain how it works.

8. (a) Define Edit Distance.

   (b) Describe some properties of Edit Distance.

   (c) Give the algorithm for finding Edit Distance.

9. (a) Define Trie, data structure for String.

   (b) Give the way of implementing vertices.

   (c) Describe the properties of Trie.

10. Give algorithm and state the runtime of search operation of Trie.

11. Give algorithm and state the runtime of insert operation of Trie.

12. Give algorithm and state the runtime of delete operation of Trie.

13. (a) Define PARTRICIA-Tree

    (b) Define Suffix Tree

    (c) Give the way and runtime for construction.

    (d) Give the space complexity for construction.

14. Give the way and runtime for efficient string matching using suffix tree.

15. Give the way and runtime for searching for many strings in one text using suffix tree.

16. Give the way and runtime for searching one string P in many texts $\mathrm{T}_1, T_2, \ldots, T_l$

17. Give the way and runtime for searching for the longest common substring in T_1, T_2

18. (a) Why is data compression important?

    (b) State two types of data compression.

    (c) Define 'code' for text compression and property of that.

    (d) Give some avoidance when we define code.

19. (a) Describe Huffman Coding.

    (b) How to encode and decode in Huffman Coding?

    (c) Give the runtime for construction and algorithm technique.

    (d) Give the runtime for encoding and decoding.

    (e) Why is Huffman Coding optimal?

20. (a) What is decision problem?

    (b) Define Problem in general.

21. (a) Define P and give some example of P.

    (b) Define NP and give some example of NP.

22. Why $P \subseteq NP$ and Is $P=NP$?

23. State the satisfiability problem. (SAT Problem)

24. State the Clique problem.

25. State the Vertex Cover problem.

26. State the Partition problem.

27. State the Traveling Salesman problem (TSP)

28. (a) State the coPrimes problem.

    (b) Is coPrimes NP?

29. (a) Define Graph Isomorphism.

    (b) Define $\leq_{\mathrm{p}}$.

30. (a) Define SUBSET SUM.

    (b) Define PARTITION.

    (c) PARTITOIN $\leq_{\mathrm{p}}$ SUBSET SUM.

31. (a) Define VERTEX COVER

    (b) Define CLIQUE

    (c) VERTEX COVER $\leq_p$ CLIQUE.

    (d) Prove that G has a Vertex Cover V' of size k if and only if $\bar{G}$ has a clique of size n-k.

32. (a) Prove that if $A \leq_p B$ and $B \in P$, then $A \in P$

    (b) Prove that if $A \leq_p B$ and $B \in NP$, then $A \in NP$

33. Prove that $\leq_p$ is transitive, i.e., if $A \leq_p B$ and $B \leq_p C$ then $A \leq_p C$

34. (a) State CNF form of SAT.

    (b) State k-SAT Problem.

    (c) SAT $\leq_p$ $CLIQUE$

    (d) Prove that if φ is satisfiable if and only if G has a clique of size k, k is the number of clauses of φ

35. (a) Define NP-Hard

    (b) Define NP-Complete

    (c) Give some remark for some relationship of NP-Hard Problems.

    (d) What is directed proved problem as NP-Hard?

36. (a) State the Optimization Problem.

    (b) How to solve the optimization problem?

37. (a) Give the state of the UNION-FIND data structure after the following sequence of operations, staring from singleton sets $\{1\}, \ldots, \{8\}$. Use path compression. In case of ties, always make the lower numbered root point to the higher numbered one.

$$\text{union}(1,2); \text{union}(3,4); \text{union}(5,6); \text{union}(7,8); \text{union}(2,4); \text{union}(6,8); \text{union}(4,8); \text{find}(1)$$

(b) Suppose that starting from the singleton partition $\{x_1\}, \ldots, \{x_n\}$ of some set $S = \{x_1, \ldots, x_n\}$ first some UNION- and then some FIND-Operations are carried out, altogether m operations. Suppose that union-by-height and path-compression is used. Show that the amortized runtime per operation is $\Theta(1)$, i.e., the total runtime of these operations is $\theta(m)$

Hint: Let k be the number of times the parent pointer of some vertex is accessed if it is pointing to the root of some tree. Find an upper bound on k? How often is a parent pointer accessed otherwise before it is pointing to the root again.

38. (a) Suppose that a long straight stretch of a highway has to be equipped with mobile phone base stations. The company has investigated that possible positions for stations are at kilometers $x_1, \dots, x_n$ from the beginning of the highway stretch. No point of the highway should be more than 5 km away from the nearest station. Give an algorithm to find the minimum number of stations to achieve that goal, if possible.

    (b) A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the ith customer has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time $\sum_{i=1}^{n}(time\ \ spent\ \ waiting\ \ by\ \ customer\ \ i)$. Give an efficient algorithm for computing the optimal order in which to process the customers.

39. Consider the brute-force algorithm for string-matching on alphabet Σ of size d. Assume a random text $T \in \Sigma^*$ of length n where each character occurs in each position with the same probability 1/d. let $P \in \Sigma^*$ be the pattern searched for of length m. Analyze exactly the expected number of comparisons between characters in terms of d, n and m.

40. In two-dimensional pattern recognition an $n \times n$-array T and an $m \times m$-array P of characters of an alphabet Σ are given and the task is to find all occurrences of P in T.

    (a) Design and analyze a brute-force algorithm.

    (b) How would you modify the Rabin-Karp-algorithm to solve this problem? Analyze the runtime in the worst case.

41. (a) Consider the following pattern $P \in \{a, b, c\}^*$:

    babcbaabccababc

    Apply the Boyer-Moore algorithm to P and a text $T \in \{a, b, c\}^*$ of length at least 50 of your choice, where both the bad-character- as well as the good-suffix-rule can be successfully applied at least once.

    (b) Give and analyze an algorithm to determine the array $\gamma[j]$ used in the algorithm of Boyer-Moore.

42. Describe and analyze efficient algorithms for search, insertion, and deletion in PATRICIA-trees. Suppose that edges have the space efficient labelling as shown in class.

43. Use suffix trees to design efficient algorithms for the following problems:

    (a) Find the longest substrings of a string w that occur more than once.

    (b) Find the shortest substrings of w that occur only once.

    (c) Find the longest substrings of w which are palindromes, i.e., read the same forward and backward.

44. Show that the following problems are in NP. Of which can you show that they are in P?

(a) Given a graph, does it contain a simple cycle of length 4?

(b) Set cover (SC):

Given sets (of integers) $S_1, \dots, S_n$ and $k \in N$. Are there k of the given sets whose union equals the union of all sets?

(c) Is a given natural number a the product of 2 prime numbers?

45. Show the following polynomial time reductions:

(a) subset sum $\leq_p$ PARTITION

Hint: $(a_1, \dots, a_n, b) \rightarrow (a_1, \dots, a_n, |\sum_{i=1}^{n} a_i - 2b|$

(b) Let SGI be the following problem: Given two graphs G_1, G_2. Is G_2 isomorphic to a subgraph of G_1. Show that $A \leq_p$SGI where A is one of the NP-problems given in class.

(c) SAT$\leq_p$3SAT

Hint: replace any SAT-clause $(y_1 \vee \dots \vee y_k), k > 3$ by 3SAT-clauses $(y_1 \vee y_2 \vee z_1) \wedge (\overline{z_1} \vee y_3 \vee z_2) \wedge \dots \wedge (\overline{z_{k-3}} \vee y_{k-1} \vee y_k)$ where $z_1, \dots, z_{k-3}$ are new additional variables.

46. A Boolean formula $\phi(x_1, \dots, x_n)$ in CNF is obviously satisfiable if $\phi|_{x_1=1}$ or $\phi|_{x_1=0}$ are satisfiable by some truth assignment to $x_2, \dots, x_n$. Here, $\phi|_{x_1=1}$ means the formula ϕ simplified by omitting clauses containing the literal x_1 and taking out the literal $\overline{x_1}$ from all clauses where it occurs. $\phi|_{x_1=0}$ is defined analogously.

    Once some clause contains no more literals the formula is not satisfiable, if the formula is empty (no more clauses) it is satisfiable.

    Use these properties to implement a recursive SAT-solver, i.e., a problem, that decides whether the input formula is satisfiable and, if so, produces a satisfying assignement.

47. Give True or False for each of the following statements. Justify your answers.

(1) A graph algorithm with $O(E\log V)$ running time is asymptotically better than an algorithm with $O(E\log E)$ running time for a connected, undirected graph $G(V,E)$

(2) If the depth-first search of a graph G yields no back edges, then the graph G is acyclic.

(3) If some of the edge weights in a graph are negative, the shortest path from s to t can be obtained using Dijkstra's algorithm by first adding a large constant C to each edge weight, where C is chosen large enough that every resulting edge weight will be nonnegative.

(4) Let G=(V,E) be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v1 and v2 must be a shortest path in G.

48. You have an exam with n questions. Each question I has integral point value $c_i > 0$ that requires $m_i > 0$ minutes to solve. Suppose that no partial credit is awarded in this exam. Your goal is to come up with an algorithm which, given $c_1, c_2, \dots, c_n, m_1, m_2, \dots, m_n, and\ C,$ computes the minimum number of minutes required to earn at least C points on the exam.

    (1) Let M(i,c) denote the minimum number of minutes needed to earn c points when you are restricted to selecting from questions 1 through i. Give a recurrence expression for M(i, c). (The base cases: for all i, and $c \leq 0, M(i, c) = 0;$ for $c > 0, M(0, c) = \infty$)

    (2) Give an algorithm to compute the minimum number of minutes required to earn at least C points on the exam and analyze the running time.

    (3) Explain how to extend your solution from the previous part to output a list S of the questions to solve such that the total score is at least C and the total number of minutes is minimized.

    (4) Suppose now that partial credit is given so that the number of points you receive on a question is proportional to the number of minutes you spend working on it. That is, you earn c_i/m_i points per minute on question i (up to a total of c_i points), and you can work for fractions of minutes. Give an $O(n \log n)$-time algorithm to determine which questions to solve (and how much time to devote to them) in order to receive C points the fastest. Prove the correctness of your algorithm.

49. Given 2 decision problems L_1 and L_2 in NP and $L_1 \leq_p L_2$ for each of the following statements, give one of T(true), F(false) or O(open question), and briefly justify your answer.

    (1) If $L_1 \in P, then\ L_2 \in P$

    (2) If $L_2 \in P, then\ L_1 \in P$

    (3) If $L_1 \in NPC, then\ L_2 \in NPC$

    (4) If $L_2 \in NPC, then\ L_1 \in NPC$

    (5) If $L_2 \leq_p L_1, then\ L_1\ and\ L_2\ are\ NPC$

    (6) Suppose L_2 is solvable in $O(n)$. Then L_1 is also solvable in $O(n)$

50. Give True or False.

    (1) We can use a 2-approximation algorithm for the minimum vertex cover problem as a 2-approximation algorithm for the maximum clique problem.

    (2) If one NPC problem can be solved in polynomial time, all of the NPC problem can be solved in polynomial time.

    (3) Suppose Q is in NP, but not necessarily NPC. Then, a polynomial time algorithm for 3SAT would necessarily imply a polynomial time algorithm for Q.

    (4) NPC problems can be reduced to any other NPC problem.

    (5) If SAT problem is in P, then co-NP $\neq$ P.

51. Let G=(V,E) be an undirected graph. A strongly independent set is a subset S of vertices such that for any two vertices, $u, v \in S$ there is no path of length ≤ 2 between u and v.

    Consider the following Strongly Independent Set (SIS) problem:

    Given an undirected graph G=(V,E) and an integer k, does G have a strongly independent set of size k?

    For each following question, give Yes or No or Unknown.

    (1) Is $\mathrm{SIS} \in \mathrm{P}$?

    (2) Is $\mathrm{SIS} \in \mathrm{NP}$?

    (3) Is $\mathrm{SIS} \in \mathrm{co-NP}$?

    (4) Is $\mathrm{SIS} \in \mathrm{NP-hard}$?

    (5) Is $\mathrm{SIS} \in \mathrm{NPC}$?

52. Suppose that a certain country has the coins with the following denominations $v(1) < v(2) < \cdots < v(k)$ (all integers). Given an integer n, we want to find the minimum number of coins to make n. (You can assume that $v(1) = 1$ to make any amount n)

    (1) Consider the greedy algorithm which repeatedly takes the largest coin possible. For the set of denominations $\{50,25,10,5,1\}$, prove that the greedy algorithm always gives the minimum number of coins.

    (2) Give the set of denominations for which the greedy algorithm does not give the minimum number of coins.

    (3) Give an efficient algorithm to solve the problem for any set of k different denominations. What is the running time of your algorithm?