CS205: Design and analysis of algorithms

Mid-semester examination • Maximum marks: 30

Spring 2022

Notes

- Instructor/TA will not be available for clearing doubts during the exam.
- If any assumptions are made, state them and justify.
- There are 6 questions in this question paper. Each carries 6 marks. **Answer only 5 questions. If you answer all 6, we will evaluate any 5 of them arbitrarily, it may not be the best 5.**
- If it is helpful for your answers, you can use the results we obtained in the lectures, without repeating the proofs discussed in the lectures.
- You are allowed to carry hand-written materials, but no printed materials and electronic gadgets are allowed.

Q1: For each of the statements below, state whether the statement is TRUE or FALSE and write one or two sentences to support your answer.

(i) There is a polynomial-time reduction from the VERTEX COVER problem to the 3-SAT problem.

(ii) A graph G has a clique of size k if and only if $\overline{G}$ has a vertex cover size k. (The graph $\overline{G}$ is the complement graph of G, i.e., uv is an edge in G if and only if uv is not an edge in $\overline{G}$.)

(iii) Let Q be a polynomial-time solvable problem. Then there exists a poiynomial-time verification algorithm A for Q, i.e., for every yes-instance I of Q, there exists a certificate y such that A takes input (I, y) and verifies (i.e., outputs TRUE) that I is a yes-instance of Q.

(iv) Let Q be any decision problem. Then there exists a polynomial-time reduction from Q to Q.

[1.5*4 = 6 marks]

Q2: The longest-simple-cycle problem is the problem of determining a simple cycle (no repeated vertices) of maximum length in an undirected graph. Formulate the corresponding decision version of the problem. Either prove that the decision version of the problem is NP-complete or come up with a polynomial-time algorithm to solve it. **[6 marks]**

Q3: In the CLIQUE problem, we are given with a graph G and an integer k, and the objective is to find whether G has a clique (mutually adjacent set of vertices) of size at least k or not. We have learned that the problem is NP-complete. Now, let's define a variant of it called CLIQUE-SPECIAL: Given an integer k and a graph G which does not have an independent set (mutually nonadjacent set of vertices) of size 4, find whether G has a clique of size at least k or not. Note that there is only one difference between CLIQUE and CLIQUE-SPECIAL - The input domain of CLIQUE is *the* set of all graph whereas the same for CLIQUE-SPECIAL is constrained that the input graphs for the problem has no independent set of size 4. Either prove that CLIQUE-SPECIAL is NP-complete or come up with a polynomial-time algorithm to solve it. **[6 marks]**

Q4: You are given with n distinct keys $K = \{k_1, k_2, \ldots, k_n\}$ in sorted order, i.e., $k_1 < k_2 < \ldots < k_n$. Additionally, for each key k_i there is an associated real number $w[i]$ such that $0 \leq w[i] \leq 1$. The objective is to build a binary search tree T with the keys such that $\text{cost}(T) = \Sigma_{i=1}^{i=n}((\text{depth}_T(k_i)+1) \cdot w[i])$ is minimized. The $\text{depth}_T(k_i)$ is the depth of the key k_i in the binary search tree T. For example, the depth of the root node is 0 and the depth of a child of the root node is 1, and so on. A tree T for which $\text{cost}(T)$ is minimized is known as an optimal binary search tree. For $1 \leq i \leq n$, and $i-1 \leq j \leq n$, let $e[i, j]$ be defined as the $\text{cost}(T_{i,j})$, where $T_{i,j}$ is an optimal binary search tree for $K_{i,j} = \{k_i, k_{i+1}, \ldots, k_j\}$ and the weight function $w[i : j]$ (denotes the subarray of w from index i to index j). Note that $T_{i,i-1}$ denotes an empty tree. Do the following:

(i) Come up with a dynamic programming formulation for $e[i, j]$ and briefly explain why the formulation is correct.

(ii) What is the worst-case time complexity of the algorithm obtained from the formulation? Briefly explain the calculation. No need to write the algorithm.

Note: Unlike what we discussed in the lecture, no dummy keys are involved in this problem. **[5+1 = 6 marks]**

Q5: You are given with a directed edge-weighted graph G, whose adjacency matrix is given in Table 1. There are five vertices $\{1, 2, 3, 4, 5\}$ which forms the 0^{th} row and 0^{th} column of the table. An entry w corresponding to the row index i and the column index j represents an edge from vertex i to vertex j with weight w. For example, there is an edge with weight 5 from vertex 2 to vertex 5. An entry ∞ denotes that there is no edge from vertex i to vertex j. Apply Floyd-Warshal algorithm on G and obtain D^0, D^1, D^2, D^3, D^4, and D^5 as discussed in the class. Recall that $D^i[a, b]$ denotes the shortest distance from vertex a to vertex b considering only the paths having intermediate vertices from $\{1, 2, \ldots, i\}$. **[6 marks]**

	1	2	3	4	5
1	0	∞	1	3	3
2	1	0	1	1	5
3	2	1	0	2	1
4	4	4	2	0	∞
5	2	1	∞	1	0

Table 1: Adjacency matrix of graph G mentioned in Q5

Q6: A palindrome is a string which reads the same from both the ends. For example, "wasitacatisaw" is a palindrome. A palindromic subsequence of a string S is a string $P = S[i_1]S[i_2] \ldots S[i_t]$ such that $i_1 < i_2 < \ldots < i_t$ and P is a palindrome, where $S[i_j]$ denotes the symbol in S at i_j^{th} position. For example, $P =$"wasitacatisaw" is a palindromic subsequence of the string $S =$ "whenadogchasedmeirantowardsa**catisaw**nearby". The symbols which constitute the palindrom is darkened.

You are given with a string S of n symbols. The objective is to find a longest palindromic subsequence in S. Let $p(i, j)$ denote the length of a longest palindromic subsequence of $S[i : j]$ (denotes the substring of S containing symbols from index i to index j). Do the following:

(i) Come up with a dynamic programming formulation of $p(i, j)$. Briefly explain why the formulation is correct.

(ii) What is the worst-case time complexity of the algorithm obtained from the formulation? Briefly explain the calculation. No need to write the algorithm.

[5+1 = 6 marks]