CS31005 Algorithms–II, Autumn 2022–2023

23-Nov-2022, 09:00am–12:00pm **End-Semester Test** Maximum marks: 50

[Answer all questions.]

---

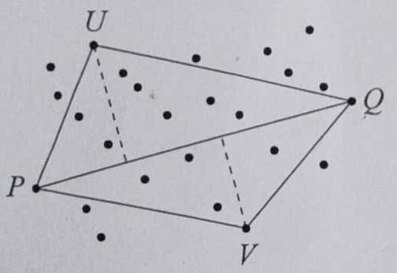
1. (a) You are given a maximum-flow algorithm A that works correctly only if (i) there are no incoming edges to the source and no outgoing edges from the sink, and (ii) both the directed edges (u,v) and (v,u) do not exist together for any pair of vertices u,v. However, you are given a graph G that violates both the above conditions. Can you still find the maximum flow in G using A? Justify your answer briefly and clearly. **(4)**

(b) Define Monte Carlo and Las Vegas algorithms. **(2)**

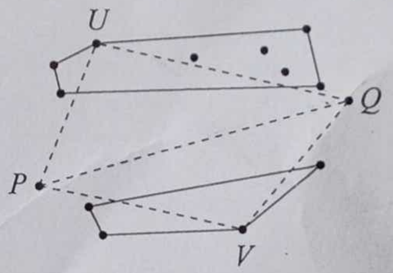
(c) Consider an application using a Bloom filter. The maximum number of entries to be inserted in the filter is 500,000, and the probability of false positives should not exceed 0.02%. What is the size of the filter (in MB, rounded up to 2 decimal places) that you will use? What will be the number of hash functions to be used? Show your calculations. You do not need to design the hash functions. **(4)**

(d) Consider the Euclidean TSP problem on a complete graph with n nodes (numbered from 1 to n), where each node is a point in the 2-d plane, and the weight of the edge (u,v) is the Euclidean distance between u and v. Suppose that you start a branch-and-bound algorithm to find the optimal tour cost starting from node 1. Consider any node u in the state-space tree (or computation tree) that represents a partial tour of $k+1$ nodes $\langle 1, x_1, x_2, \ldots, x_k \rangle$ with cost C_k, where $x_i \in \{2,3,\ldots,n\}$ for $1 \leqslant i \leqslant k < n$, and $x_i \neq x_j$ if $i \neq j$. Let the cost of the best complete tour seen so far be C_{best}. Suggest an $O(1)$-time pruning heuristic (bounding function) that you can apply to the node u to decide whether to explore the subtree rooted at u or not. For you to get any marks, your heuristic must be different from any min-weight-based heuristic, and its correctness must depend on the triangle inequality. Give a brief justification as to why it is correct. **(4)**

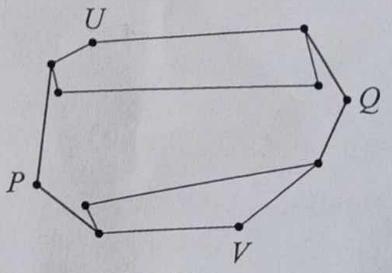
(e) We compute the convex hull H of a set S of n points in the plane using an idea described now. Assume that the given points are in general position. We first compute the leftmost point P and the rightmost point Q in S. We also identify the most distant (perpendicular distance) points U and V on the two sides of the line PQ. See Part (a) of the figure below. The points inside the triangles PQU and PQV cannot lie on the convex hull, so we remove these points from S. We then recursively compute the convex hull H_1 of the subset of points of S, that lie above PQ. We also recursively compute the convex hull H_2 of the subset of points of S, that lie below PQ. See Part (b) of the figure below. Finally, we compute appropriate tangents from P and Q to the hulls H_1 and H_2 (Part (c) of the figure below) to get H.



(a)



(b)



(c)

Write, with clear justifications, the recurrence for the worst-case time complexity $T(n)$ of this algorithm, and deduce that $T(n)$ is $O(n^2)$. **(4)**

2. Foomazon Logistic Company serves n customers. Its items are dispatched from $t \geqslant 2$ warehouses. Each warehouse has a vehicle. Each vehicle starts from its warehouse, serves a subset of customers (each customer only once), and comes back to its warehouse. A cost (a positive integer, assumed symmetric) is given to travel from one customer to another, and also from each warehouse to each customer. Foomazon wants to schedule its vehicles to serve disjoint subsets of customers in such a way that the total cost of serving all the customers is minimized. Note that each customer is to be served exactly once by exactly one vehicle.

(a) Foomazon faces an optimization problem. Propose an equivalent decision version of the problem, and show that the optimization problem can be solved in polynomial time if and only if the decision problem can be solved in polynomial time. **(4)**

(b) Prove that the decision problem of Part (a) is NP-complete. Use (the decision version of) TSP in your reduction, but note that $t \geqslant 2$ in the Foomazon problem. **(6)**

3. Let $G = (V, E)$ be an undirected graph. For two subsets X, Y of V (not necessarily disjoint), denote by $N(X, Y)$ the number of edges of G with one endpoint in X and the other in Y. If X contains only a single vertex x, we also write $N(x, Y)$ to denote the number of neighbors of x in Y. A 3-cut of G is a partition of V into three mutually disjoint non-empty subsets U_1, U_2, U_3. The size of the 3-cut U_1, U_2, U_3 is $C(U_1, U_2, U_3) = N(U_1, U_2) + N(U_2, U_3) + N(U_3, U_1)$. The MAX-3-CUT problem deals with finding a 3-cut U_1, U_2, U_3, for which $C(U_1, U_2, U_3)$ is as large as possible.

Let $v_1, v_2, v_3, \ldots, v_n$ be an arbitrary ordering of the vertices of G. Consider the approximation algorithm Q3 for solving the MAX-3-CUT problem. Assume that $n \geqslant 3$.

Algorithm Q3

```
Initialize $U_1 = \{v_1\}$, $U_2 = \{v_2\}$, and $U_3 = \{v_3\}$.
For ($i = 4$; $i \leqslant n$; $i$++) repeat:
    Find the minimum of $N(v_i, U_1)$, $N(v_i, U_2)$, and $N(v_i, U_3)$.
    If the minimum is achieved by $U_j$ (ties are broken arbitrarily), add $v_i$ to $U_j$.
Return $U_1, U_2, U_3$.
```

(a) Find and prove an invariance (an inequality involving the counts $N(U_i, U_j)$) of the loop in Algorithm Q3. (An invariance of the loop is a statement that is true at every instant when the loop condition $i \leqslant n$ is checked.) **(4)**

(b) Using the invariance of Part (a), establish that Algorithm Q3 is a $\frac{2}{3}$-approximation algorithm. **(2)**

(c) Prove that this approximation ratio is tight. Your proof must use an *infinite* family of *connected* graphs. **(4)**

4. (a) Consider an undirected graph $G = (V, E)$ with $|V| = n$, and with a positive weight $w(v)$ assigned to each vertex v of the graph. Let Δ be the maximum degree of a vertex in G. An independent set of the graph is a subset S of V such that no two vertices in S are adjacent to each other in G. The weight of an independent set is the sum of the weights of the vertices in that set. The *maximum weighted independent set problem* is to find an independent set in G with the maximum weight. The problem is known to be NP-hard. The greedy randomized algorithm Q4 is proposed for solving this problem.

Algorithm Q4

```
Choose a random permutation $v_1, v_2, \ldots, v_n$ of the vertices.
Initialize $S = \emptyset$.
For $i = 1, 2, 3, \ldots, n$ (in that order), repeat:
    If no vertex in $S$ is adjacent to $v_i$, then set $S = S \cup \{v_i\}$.
Return $S$.
```

For any $u \in V$, derive a lower bound on the probability that u is chosen in S. Your bound should be expressed in terms of Δ, and must be independent of the number of nodes, the number of edges, and the vertex weights. Give a clear justification (no marks without it). **(4)**

(b) Show that the expected weight of the independent set S returned by Algorithm Q4 is at least $\frac{1}{\Delta+1}$ times the optimal (the weight of the actual maximum-weight independent set). To show this, first define, for each $u \in V$, an indicator variable X_u which is 1 if u is included in S, 0 otherwise. Then use these variables to do the rest. Your proof must start with these indicator variables. **(4)**

(c) Consider a Tweeter-like system used by 1 million (10^6) users. Each user's tweets are stored as a record of the form (username, collection of tweets). Given the large number of users, the data is divided into five servers, each containing the records of 200,000 users (one user's data is stored in exactly one server). A user can log in to any of the five servers, henceforth referred to as the log-in server. If the user's data are not stored in the log-in server, his/her tweets need to be brought from the server they are stored in. You should ensure that most of the time, the log-in server communicates with at most one other server (the correct one) to get the user's data. Propose a scheme based on Bloom Filters, to achieve this (no marks if Bloom filters are not used). In particular, clearly specify

(i) exactly what Bloom filter(s) is/are kept in each server, and **(2)**

(ii) the actions taken by the log-in server to get a user's data when that user logs in. **(2)**