CS31005 Algorithms – II, Autumn 2021–2022

Third Test

Nov 16, 2021, 2:15pm — Maximum marks: 45

Duration: 80 minutes (65 minutes for answering questions + 15 minutes for download/submission)

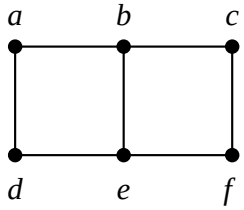
All your answers MUST BE HANDWRITTEN on paper. Scan all papers with your answers in a SINGLE pdf, and upload it as the answer to the Quiz in Moodle. The size of the final pdf must be less than 10 MB. You must upload the pdf strictly by 3:35 pm Moodle server time, the quiz submission will close after that.

Answer all questions.

1. Consider the following algorithm for computing the maximum (unweighted) cut in an undirected graph $G = (V, E)$. Here, $S - u$ means remove u from S, and $S + v$ means add v to S. Likewise, for $T + u$ and $T - v$.

```
Start with an arbitrary cut (S,T) of V.
while (1) {
    If for some u ∈ S, moving u from S to T increases the cut size, then set (S,T) = (S−u, T+u),
    else if for some v ∈ T, moving v from T to S increases the cut size, then set (S,T) = (S+v, T−v),
    else break;
}
Return (S,T).
```

This algorithm is run on the following graph with the initial cut $S = \{a, b, c\}$ and $T = \{d, e, f\}$.



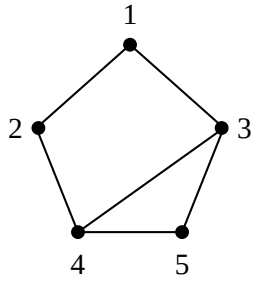
Explain the workings of the iterations of the above algorithm in the format given below. **(8)**

			Number of neighbors in own part						Number of neighbors in other part					
	S	T	a	b	c	d	e	f	a	b	c	d	e	f
Initialization	$\{a,b,c\}$	$\{d,e,f\}$												
After Iteration 1														
After Iteration 2														
⋮														

2. Consider an undirected graph $G = (V, E)$ with n nodes numbered $1, 2, 3, \ldots, n$. We want to find a maximum independent set S of G (the maximum sized subset S of V such that there is no edge between any two nodes in S). The following Branch-and-Bound algorithm is proposed for it (notations used are from slides given).

- Structure of solution S : $\langle x_1, x_2, x_3, \ldots, x_n \rangle$ with $x_i = 1$ if node i is in S, 0 otherwise.
- $F(S) = \sum_{k=1}^{n} x_k$.
- Initial value of v_{lower} (lower bound of optimal value) = 1.
- For any partial solution $S_i = \langle x_1, x_2, x_3, \ldots, x_i \rangle$, $i < n$:
  - $F(S_i) = \sum_{k=1}^{i} x_k$.
  - S_i is feasible if for any x_j, x_k, $0 < j, k \leqslant i$, $j \neq k$, if $x_j = x_k = 1$ then $(x_j, x_k) \notin E$.
  - $v_{upper}(i+1) = (n-i)$ (that is, if all remaining nodes are included in the independent set).
- Order of traversal is DFS with $x_i = 1$ generated first (left child) for each i.
- A live node is expanded if and only if it represents a feasible partial solution and there is a chance of finding a better optimal solution starting from it.

Show the complete state-space tree that will be generated if the above algorithm is applied on the following graph. **(7)**



3. Ms. Trotter wants to make a sightseeing tour to n locations. She obtains the cost $c_{i,j}$ of traveling from Location i to Location j for all i, j. She plans to visit each sightseeing location at most once, and eventually come back to the location from which she starts. She however has a limited budget B, and can afford a tour if and only if the total cost of the tour is no more than B. She needs to identify the maximum number m of locations she can visit subject to her budgetary constraint. You may assume that all $c_{i,j}$ and B are positive integers, and that $c_{i,j} = c_{j,i}$ for all i, j. The costs need **not** satisfy the triangle inequality.

(a) Ms. Trotter's problem is an optimization (maximization) problem. Propose an equivalent decision version of the problem. You need to prove that the optimization problem can be solved in polynomial time if and only if the decision problem can be solved in polynomial time. **(5)**

(b) Prove that the decision version of Ms. Trotter's problem is NP-Complete. **(5)**

4. Let $A = (a_1, a_2, \ldots, a_n)$ be an array of n positive integers, and t a target sum (a positive integer again). The task is to find a subset $I \subseteq \{1,2,3,\ldots,n\}$ for which the sum $\sum_{i \in I} a_i$ is as small as possible but at least as large as t. Assume that $t \leqslant \sum_{i=1}^{n} a_i$ (otherwise the problem has no solution). Prof. Sad proposes the following algorithm to solve this problem.

```
Initialize sum = 0, and I = ∅.
for i = 1,2,3,...,n (in that order), repeat {
    Update sum = sum + a_i, and I = I ∪ {i}.
    If sum ⩾ t, break.
}
Return I.
```

(a) Prove that the approximation ratio of Prof. Sad's algorithm cannot be restricted by any constant value. **(3)**

(b) Prof. Atpug suggests sorting the array A in the ascending order before running the algorithm. In view of this suggestion, we now have $a_1 \leqslant a_2 \leqslant a_3 \leqslant \cdots \leqslant a_n$. Prove that Prof. Atpug's suggestion makes Prof. Sad's algorithm a 2-approximation algorithm. **(7)**

5. **(a)** Consider a group of 1000 people, divided into 10 groups, with each group containing exactly 125 people. Note that this means some people will appear in more than one group. It is also given that at least one person is different between any two groups. We wish to choose a group of people (from the 1000 people) as leaders so that there is at least one leader and at least one non-leader in each group. Propose a Monte Carlo algorithm for the problem with an error bound less than $1/2^{10}$. Briefly justify your answer. **(3)**

(b) Consider a document-storage system. Each document is identified with a set of keywords. The documents are stored in two servers X_1 and X_2, with X_1 holding all documents containing any keyword from a set of keywords K_1, and X_2 holding documents with any keyword from a set of keywords K_2. The sets K_1 and K_2 are disjoint. However, note that since a document can contain keywords from both K_1 and K_2, the set of documents stored by the servers need not be disjoint.

A user queries for documents containing a set of keywords K to a central server C. Assume that K can contain keywords from only K_1, only K_2, or from both K_1 and K_2. Server C returns all documents from X_1 and X_2 that contain ALL of the keywords in K. To do this, C sends the keywords in K to X_1 and X_2, and they communicate to send the final set of documents to C to be sent back to the user. However, you cannot send any keyword from set K_2 to X_1, or any keyword from set K_1 to X_2 at any stage. Your goal is to reduce the number of bits transferred between the servers. Suggest a Bloom filter based scheme to do this. Note that the final answer should be fully correct (no false positives). You need not try to reduce computation at a server or total time taken. For the bloom filter used, you do not have to design any hash functions exactly, just say what it will contain, and how it will be used.

List all the steps clearly, starting from C receiving the query from the user till C sending the response to the user. **(7)**
