CS31005 Algorithms–II, Autumn 2022–2023

Class Test 1

09–September–2022 07:00pm–08:00pm Maximum marks: 20

Roll no: ____________________ **Name:** ____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions. If you use any algorithm/result/formula covered in the class, just mention it, do not elaborate.*]

1. Let $i = (b_{l-1}b_{l-2}\ldots b_2b_1b_0)_2$ be a positive integer. Denote the position of the least significant one-bit of i by $\text{pls1}(i)$. That is, if $b_0 = b_1 = b_2 = \cdots = b_{t-1} = 0$ and $b_t = 1$, then $\text{pls1}(i) = t$. Consider the following code.

```
i = 1;
while (i <= n) {
   ++i;
   f(i);
}
```

Assume that the initialization `i = 1`, the check `i <= n`, and the increment `++i` can each be done in one unit of time. However, the function call `f(i)` takes $t^2 + 5$ units of time, where $t = \text{pls1}(i)$. Since $t = O(\log n)$, we have a worst-case bound of $O(n \log^2 n)$ for the running time of the above code snippet. In this exercise, we use an amortized analysis to arrive at a better bound.

(a) Define a potential function as follows. Let i have exactly k one-bits at positions $p_1, p_2, \ldots, p_k$ (positions are counted from the least significant, that is, right end starting from 0). Define the potential for this i as $\Phi_i = (2p_1 + 3) + (2p_2 + 3) + \cdots + (2p_k + 3)$. Prove that this is a valid potential function. **(2)**

Solution The initial potential is $\Phi_1 = 3$. For any $i \geqslant 1$, we have at least one one-bit at some position $p \geqslant 0$. Therefore $\Phi_i \geqslant (2p+3) \geqslant 3 = \Phi_1$.

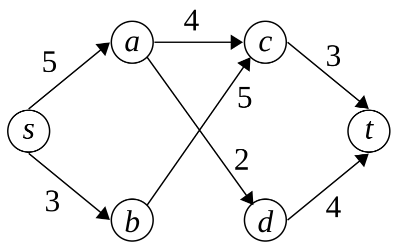
(b) Prove using the potential function of Part (a) that the amortized cost of each iteration of the loop is $O(1)$. Show your calculations. **(4)**

Solution In an iteration with $\text{pls1}(i) = t$ (after the increment), the potential reduces by

$$[3 + 5 + 7 + \cdots + (2t+1)] - (2t+3) = [(t+1)^2 - 1] - (2t+3) = t^2 - 3.$$

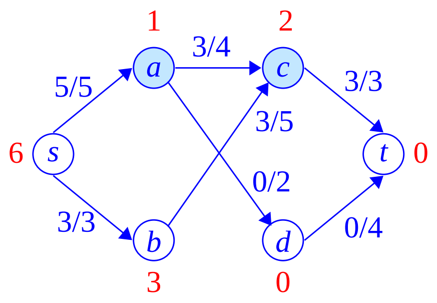
Therefore the amortized cost of that iteration is $(t^2 + 7) - (t^2 - 3) = 10 = O(1)$.

2. Consider the following flow network with source s and sink t and with the edge capacities shown against the edges.



We use the push-relabel algorithm to compute a maximum flow in this network. Suppose that after the initialization step, the first four pushes are made along the edges (a,c), (c,t), (c,a), and (b,c). Relabeling is done before a push if necessary. Redraw the network after these four pushes. Clearly show the preflow and the labels (that is, heights) of all the vertices at this instant. Also identify the active (that is, overflowing) vertices. There is no need for any explanation. Neither is there any need to show the residual network. **(4)**

Solution The preflow and the capacity are shown as $f(e)/c(e)$. The heights are shown against the vertices. The active vertices are shaded.



3. We run the Gale–Shapley algorithm for four men m_1, m_2, m_3, m_4 and four women w_1, w_2, w_3, w_4. Assume that men propose, and women take decisions (accept, reject, or replace). Give an explicit example of the preference lists for which the algorithm pairs none of the four men with their respective first choices. Do not just argue that such an example is possible. Write only two 4×4 preference tables. There is no need for explaining how the Gale–Shapley algorithm works on your preference tables. **(5)**

Solution Consider the following preference tables.

m_1	w_1	w_3	w_2	w_4
m_2	w_2	w_4	w_1	w_3
m_3	w_1	w_2	w_3	w_4
m_4	w_2	w_1	w_3	w_4

w_1	m_4	m_2	m_1	m_3
w_2	m_3	m_1	m_2	m_4
w_3	m_1	m_2	m_3	m_4
w_4	m_2	m_1	m_3	m_4

Suppose that men propose in the round-robin fashion (it does not matter though, because the Gale–Shapley algorithm outputs *the* man-optimal matching irrespective of the order of the proposals). In the first round, the provisional engagements (m_1, w_1) and (m_2, w_2) are made, and m_3 and m_4 stay unengaged. Then, m_3 proposes to w_2 and m_4 to w_1. The two provisional engagements of the first round are replaced by (m_4, w_1) and (m_3, w_2). This leaves both m_1 and m_2 unengaged again. They propose to their second choices and are both accepted. The final matching produced is $\{(m_1, w_3), (m_2, w_4), (m_3, w_2), (m_4, w_1)\}$.

Although not asked in the question, it is interesting to note that in this example, all men get their second choices, and all women get their first choices.

4. The gymkhana secretary wants to assign its tennis court to practice matches between two rival clubs. The first club has m players $X_1, X_2, \ldots, X_m$, and the second club has n players $Y_1, Y_2, \ldots, Y_n$. No match can be scheduled between players of the same club. If some X_i wants to play with some Y_j, then X_i registers a request to the gymkhana secretary. If Y_j too registers a request for playing with X_i, a match between X_i and Y_j can be scheduled. To give everyone a chance, the secretary wants to ensure that no member from either club plays more than ten matches. Moreover, for each registered pair (X_i, Y_j), there can be at most three matches. How should the secretary finalize the set of practice matches to be scheduled such that the total number of matches played is maximized? Your algorithm should output the exact set of matches (who plays who and how many times) to be played. **(5)**

Solution Create a directed graph with $m+n+2$ nodes: a set of nodes $X_1, X_2, \ldots, X_m$, a set of nodes $Y_1, Y_2, \ldots, Y_n$, and two nodes s (the source) and t (the sink). Add an edge from s to each X_i and an edge from each Y_j to t, each with capacity 10. If two players X_i and Y_j both wish to play with each other, add an edge from X_i to Y_j with capacity 3. Now, solve the maximum-flow problem. The (X_i, Y_j) edges with non-zero flow identify the pairs who play together and how many times they play.

For rough work