Tutorial-1

1. Consider a stack with a fixed size K with PUSH, POP operations (assume PUSH operations fail in O(1) time if stack is full and POP operations fail in O(1) time if stack is empty). Which of the following are NOT valid potential function for amortized analysis of a sequence of n PUSH and POP operations (choose all that apply)?
    a. The number of elements in the stack
    b. The number of free spaces in the stack
    c. The number of free spaces - the number of elements in the stack
    d. The number of free spaces + the number of elements in the stack

    [Solution Sketch] Consider a sequence of PUSHs. Options b and c violate the $\phi(D_i) \geq \phi(D_0)$ condition for a potential function

2. A sequence of stack operations is performed on stack whose size never exceeds k. After every k operations, a copy of the entire stack is made for backup purposes. Show that the cost of n stack operations, including copying the stack, is O(n) using the accounting method. Consider both cases when the element are no longer in the stack after the copy and when the elements are left in the stack after the copy.

    [Solution Sketch] There can be two variations. In variation 1, when the copy of the stack is made, the elements are no longer in the stack after the copy. In this case, the copy is like a pop only (with some additional work). In this case, you can include the cost of the pop (if it happens before copy) and copy in Push. So for accounting method, start with amortized cost of Push = 2, Pop = 0, Copy = 0. For a push, of the 2, one is for the push itself, one pays for popping it if done, or copying it to backup if it is not popped. Note that an element can either be popped or copied to backup but not both in this variation.

    In the other variation, when the copy of the stack is made, the elements are left in the stack after the copy. So in this case, simple solution is to start with amortized cost of Push = 2, Pop = 2, and Copy = 0. As the copy is made after every k operations and each copy can copy at most k elements (as stack size never exceeds k), each push/pop operation pays one extra cost for the copy if it needs to be done.

3. Consider the implementation of a queue using two stacks A and B (I am sure you all know this implementation). Find the amortized cost of a sequence of n enqueue and dequeue operation using each of aggregate, accounting, and potential method.

    [Solution Sketch] The implementation is probably known to all of you, but here it is again anyway. Keep 2 stacks, IN and OUT. A push simply pushes in the IN stack. A pop can have two cases: (i) if OUT stack is empty, pop everything from the IN stack and push it in OUT stack one by one, then pop the top element from the OUT stack, (ii) if OUT stack is not empty, just pop the top element from the OUT stack.

Aggregate method: Every element can be (i) pushed at most twice, once in the IN stack and once from the IN to OUT stack, and (ii) popped at most twice, either from the OUT stack or for transferring from IN to OUT stack. So total time = O(n), amortized cost O(1) per operation.

Accounting method: Break the two operations into three operations: (i) Push, (ii) Easy dequeue (when OUT is not empty), (iii) Hard dequeue (when OUT is empty). Give amortized costs Push = 3, Easy dequeue = 1, Hard dequeue = 1. Basically easy dequeue pays for itself (just a pop, does not generate or use a credit), every push stores two credits for the hard dequeue for the move, one for popping it from IN and one for pushing it to OUT. The hard dequeue uses the credits stored with each element pushed to pop it from IN and push it to OUT, plus one final pop. So all O(1).

Potential method: Choose potential = no. of elements in IN stack. Push increases potential by 1, easy dequeue leaves it unchanged, and hard dequeuer decreases it by –k, where k is the no. of elements in IN stack when the hard dequeue is done. So Amortized cost of push = O(1) + 1 = O(1), of easy dequeuer = O(1) + 0 = O(1), of hard dequeue = O(k) – k = O(1).

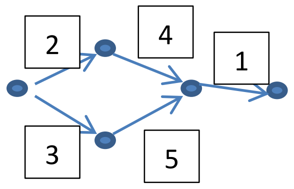
4. Suppose a graph, in addition to capacities on edges, has a capacity on every vertex such that the total inflow (and therefore the total outflow from the node) cannot exceed the capacity of the vertex. The other constraints remain the same as in the standard maximum-flow problem. Can you find the maximum flow in this graph?

   [Solution]:

   - Replace each vertex v with two vertices v1 and v2, with a directed edge from v1 to v2 with capacity equal to the vertex capacity.
   - For each edge (u, v), replace (u, v) with (u,v1) with same capacity
   - For each edge (v, w), replace (v, w) with (v2, w) with same capacity
   - Now you have a graph with only edge capacities. Find the maximum flow in it.

5. Suppose all capacities in a graph are distinct? Will the max-flow be unique?

   [Solution]: No. Consider the following graph. It is obvious that the max flow is not unique (can use either the 2-4 capacity path or the 3-5 capacity path to achieve the max flow of 1.



6. A company running a factory has to arrange for extra maintenance staff to work over vacation periods. There are K vacation periods in the year, numbered from 1 to K, with the i-th vacation period having d(i) days. There are N maintenance staff available numbered from 1 to N, with the j-th maintenance being available for a total of c(j) days. For each vacation period i, each staff can only work on a subset of the d(i) days (for example, if a vacation period spans over Friday, Saturday and Sunday, staff 1 may be available for only Friday and Saturday but not Sunday, staff 2 may be available for only Friday and Sunday but not Saturday, and staff 3 may be available for all days). What is the maximum no. of vacation days that can be covered by the company with its available staff?

   [Solution]: Create a bipartite graph with two partitions X and Y. X contains one node for each maintenance staff (say node j for staff j). Y contains one node for each vacation day of each vacation period (basically all vacation days, so if you have 3 vacation periods with 2, 4, and 3 days respectively, you will have 2+4+3 = 9 nodes in Y). Add an edge from node j in X to node i in Y if maintenance staff j can work on the vacation day represented by node i.
   Now add two vertices s and t. Add an edge from s to each node j in X with capacity c(i). Add an edge from each node i in Y to t with capacity 1.
   Now find the maximum flow from s to t. The value of the flow is the required answer.

7. Consider a p × q matrix X. Each element of X is a non-negative real number. However, the sum of the elements in any row and in any column is an integer (may be different for different rows and columns). Construct a p × q matrix Y such that each element of Y is a non-negative integer, and the sum of the elements in any row and in any column in Y is the same as the sum in the corresponding row and column respectively in X. You must use maximum flow concepts to construct Y.

   [Solution]: Let R[i] denote the sum of the elements of row j and C[j] denote the sum of the elements of column j in the matrix X.

   Create a graph with the following nodes and edges:

   - Add a set of nodes P = {a_1, a_2, …a_p}, one for each of the p rows of the matrix
   - Add a set of nodes Q = {b_1, b_2, …b_q}, one for each of the q columns of the matrix
   - Add two other nodes s and t
   - For each a_i in P, add an edge from s to a_i with capacity R[i]
   - For each b_j in Q, add an edge from b_j to t with capacity C[j]
   - For each a_i in P, add edges to every b_j in Q with capacity ∞

   Now compute the maximum flow in this graph. Since all capacities are integers, there exists an integral flow. The flow assigned on the edge (a_i, b_j) for each (i, j)-pair will give the value of the (i, j)-th element of Y.

8. A college has N students $X_1, X_2, \ldots, X_N$, M departments $D_1, D_2, \ldots, D_M$, and P societies $S_1, S_2, \ldots, S_P$. Each student is enrolled in exactly one department, and is a member of at least one society. The college has a student association (like your gymkhana) with one member from each society (You can assume that every society has at least one member). However, the society members have to be chosen such the student association has at most Q_k members from any department D_k. Design an algorithm to answer (yes or no) whether such a student association can be formed or not given the society memberships and the departments of the students. You must model the problem as a maximum flow problem.

[Solution]: Create a graph as follows:

- For each society S_i, add a node $V(S_i)$
- For each student X_j, add a node $V(X_j)$
- For each department D_k, add a node $V(D_k)$
- Add a source node s and a sink node t
- Add an edge from s to each node $V(S_i)$ with capacity 1
- Add an edge from a node $V(S_i)$ to a node $V(X_j)$ with capacity 1 if the student X_j is a member of society S_i
- Add an edge from a node $V(X_j)$ to a node $V(D_k)$ with capacity 1 if the student X_j belongs to department D_k
- Add an edge from each node $V(D_k)$ to t with capacity Q_k

Compute the maximum flow in this graph. If the maximum flow value is = P, the number of societies, then the answer is yes.