A3 ICCS 312 Algorithms and Tractability

ALEXANDER OGAY 6380727

June 2023

1 Problem 1

(a) Find the maximum flow f and a minimum cut.

The maximum flow is 11. The minimum cut is (A, C), (A, D), (B, D), (B, C)

(b) Draw the residual graph G_f (along with its edge capacities). In this residual network, mark the vertices reachable from S and the vertices from which T is reachable.

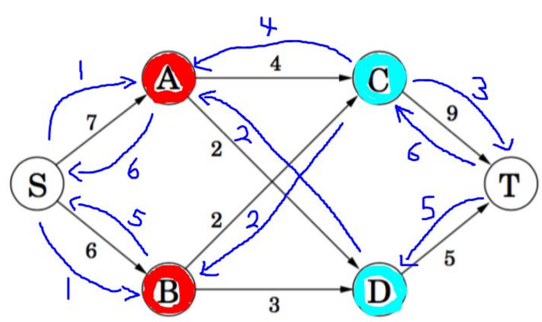

Figure 1: Residual Graph

Red, vertices reachable from S. Blue, vertices reachable from T.

2 Problem 2

Suppose Dijkstra's algorithm is run on the following graph, starting at node A. Draw a table showing the intermediate distance values $\pi[.]$ of all the nodes at each iteration of the algorithm. You must also indicate the working node at each iteration.

Iteration		0	1	2	3	4	5	6
Working Node		-	A	B	C	D	E	F
π[.] of all the nodes	A	0	0	0	0	0	0	0
	B	∞	4	4	4	4	4	4
	C	∞	15	15	13	8	8	8
	D	∞	∞	5	5	5	5	5
	E	∞	∞	25	25	12	12	12
	F	∞	∞	∞	7	7	7	7

Figure 2: Table

3 Problem 3

A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the i-th customer has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time $T = \sum_{i=1}^{n}$ (time spend waiting by customer i). Give an efficient algorithm for computing the optimal order in which to process the customers.

To minimize waiting time, we need to sort the customers by the service time that they require, starting with the customers who need the least serving time to those that need the most serving time. We can use mergesort to sort the customers in ascending order. This allows for the fastest sorting time.

4 Problem 4

You are driving down a very long highway, with gas stations at mile-posts $m_1, m_2, ..., m_n$, where $m_1 = 0$ is your starting point and m_n is your final destination. You want to make as few gas stops as possible, but your car can only hold enough gas to cover M miles. Give an algorithm to find the minimum number of stops you need to make. Argue the correctness of the algorithm, and analyze its running time. (Hint: you need to show that your strategy is at least as good as any other optimal solution.)

We make a for loop that goes through each mile-post and subtracts the gas required to make it to that mile-post, each time it hits a mile-post it checks if there is gas ≥ 0. If yes, it goes to the next mile-stop if not, it adds a mile-stop and resets the gas to full volume. This means that the algorithm runs in O(n) time. This solution works

because it checks all the mile-posts only once and it can account for the mile-posts having different distances from each other.

5 Problem 5

A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S . For instance, if S is 5, 15, -30, 10, -5, 40, 10, then 15, -30, 10 is a contiguous subsequence but 5, 15, 40 is not. Give a linear-time algorithm for the following task:

Make a for loop that goes through each number and it keeps track whether the next number is lower or higher than the number before it, as the numbers alternate between being bigger or smaller, it keeps track of the current contiguous sum and replaces it whenever there is a higher sum available.

6 Problem 6

You are working with an architect to build a brick stair. The rule of building a stair is as follows: you are given n bricks, each having equal height. You can stack bricks together to form a higher step, for example, 2 bricks to form a step of height 2. It is required that a stair must consist of at least 2 steps and that the height of the subsequent step must be strictly lower than the current one (that is the stair is descending). Also, a step must consist of at least a brick, and lastly, you must use all of the given n bricks to form a valid stair. Being a lazy person, instead of designing the stair by himself, the architect asks you to find all combinations of the possible stair instead. Give an efficient algorithm to find the number of all of the possible combinations of valid stairs that can be constructed using n bricks where $n \geq 3$.

We can make a loop that goes through each n-1 number in descending order and have a running total that adds 1 each time it loops, combining those numbers gives us a combination and we recursively check the running total to see whether it makes another stair combination, this ensures we get all the combinations possible at each stage of going through the loop n-1 times.

7 Problem 7

Suppose that you can perform the following three operations to a number n: add 1, subtract 1, and "divide by 2" if n is even. Then, given an integer n, give an efficient algorithm using the combination

of the operations given above to find the amount of least number of operations needed to turn the number into 1 where $n \geq 2$.

We check whether the number is even or odd, then we subtract one and divide, each division we check whether the number is even or odd until the number reaches 1, and keep track of each operations.