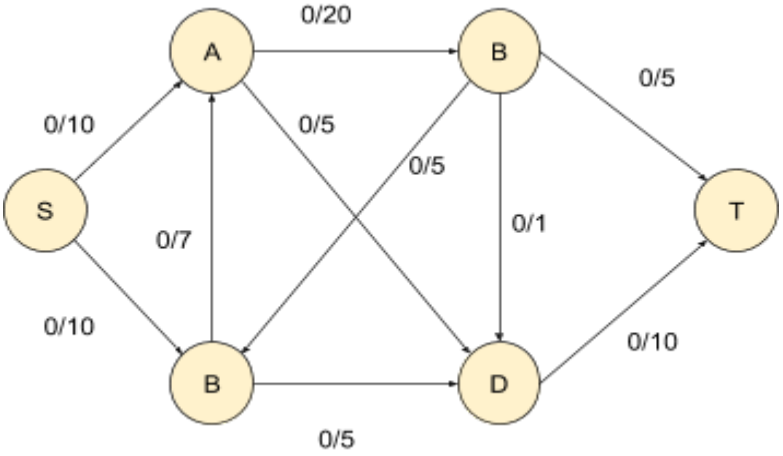
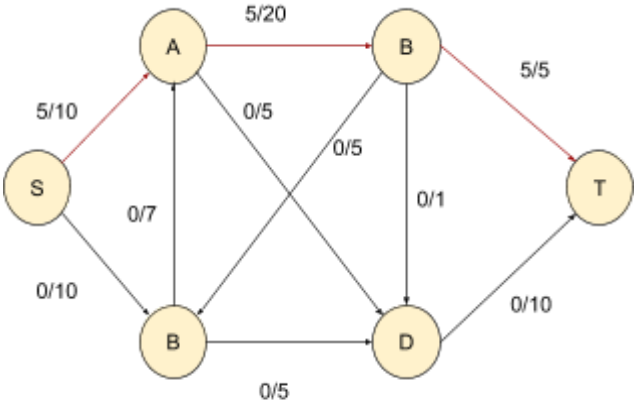
P.K.K.D. Kankanamge
140284J

Advanced Algorithms
Assignment -02

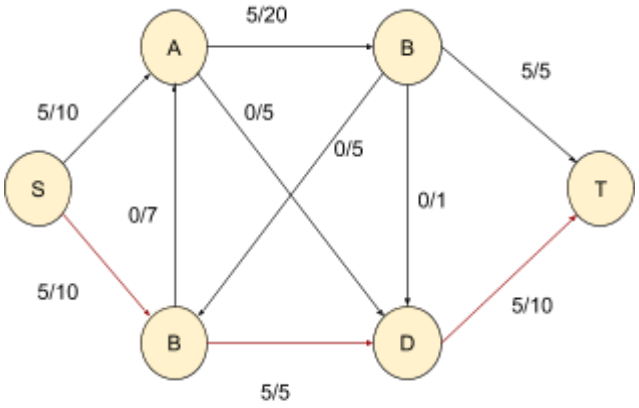
---

1. a.

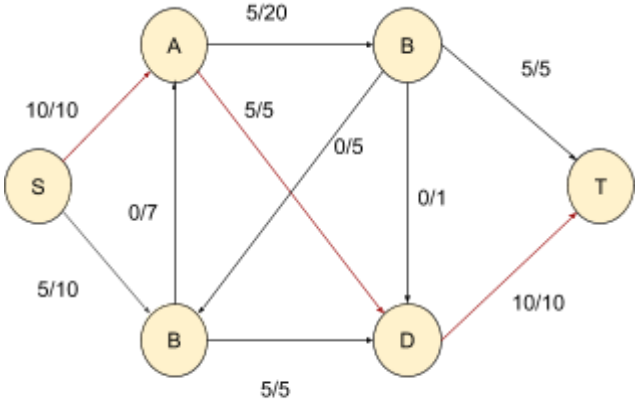




Cf(P) = min(10, 20, 5) = 5



Cf(P) = min(10, 10, 5) = 5

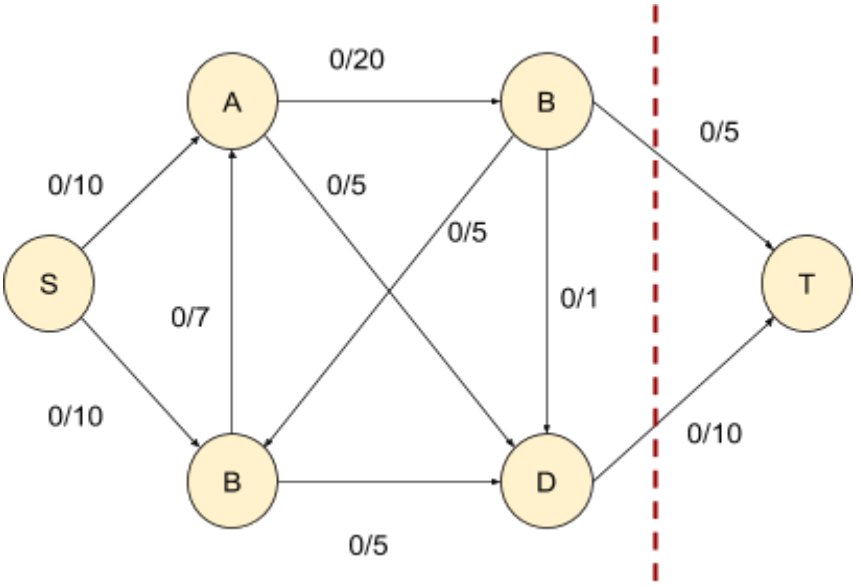


Cf(P) = min(10, 5, 5) = 5

From the following flow graphs
It is clear that no augmenting paths are
Available after this point.

Therefore max flow = 5 + 5 + 5
= 15

b.



2.

a. Parallel for i = 1 to n:

$b_i = 0$

for j = 1 to n:

$b_i = b_i + a_{ij}$

Outer loop iterates through each row while the inner row computes the sum. Parallelization is done only on the outer loop because, if inner loop is parallelized, a race condition will occur because it uses b_i which will be a shared resource once parallelized.

b. Parallel_Mat_Compute (A, B, n, i, j):

if (i == j):

b_i =0

for k= 1 to n

$b_i = b_i + a_{ik}$

else:

mid = (i + j) / 2

Spawn Parallel_Mat_Compute (A, B, n, i, Mid)

Parallel_Mat_Compute (A, B, n, Mid + 1, j)

Sync

Assumption - n is a power of 2

Two processed will be spawned and the first process which is newly created will compute the sum for the elements starting from first element upto mid element while

the main process calculates sum from mid element to last element. Then the result will be sync.

c.

$T_1 = 2T(n/2) + N$

$T_\infty = T_\infty(n/2) + \Theta(N)$

d.

$N = n$

$T_1 = 2T(n/2) + N$

$T_1 = \Theta(n * n) = \Theta(n^2)$

$T_\infty = T_\infty(n/2) + \Theta(N)$

$T_\infty = \Theta(\log_2 n) + \Theta(N) = \Theta(n)$

Parallelism $= T_1 / T_\infty$

$= \Theta(n)$

3.

a. Strategy 1 is not a good strategy. In worst case this strategy is too expensive. Consider the following example,

Assume Sequence of operations Push -> Pop -> Push -> Pop when array is full at L/2. Each of these operations take a time proportional to k which is the size of the array.

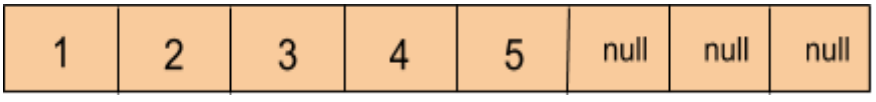
Let the following be the initial array,

K = 5

L = 8

Perform a Pop Operation and array gets reduced by half as shown below.

And then a Push operation is performed making the array double in size again,



This making the array size double and then reducing as shown above happens each time when pushing and popping is done in sequence.

b. By the same argument mentioned above, reducing the array when its L/4 is more effective. This is shown below,

Let initial array be,

K = 5

L = 8

Same Pop -> Push -> Pop sequence of operations are applied but the array size does not change.

This changes only when L/4

4.

Array Size	Binary Search (μs)	Randomised Search (μs)
10^5	2.8290748596191406	11.23568725585938
10^6	3.7872791290283203	15.835285186767578
10^7	4.352331161499023	17.85564422607422

Amortized cost of binary search when compared with the amortized cost of Randomised Binary search is less. This is because binary search takes only the size into account while the randomised search takes size of the input as well as the output from a random number generator.

Binary search guarantees that input array is divided into two by the mid element while in the worst case, Randomised search will have to look from the very first element if its index is returned by the random number generator.

This makes Randomised Binary Search to take more time when compared with the Binary Search.