SUMMARY

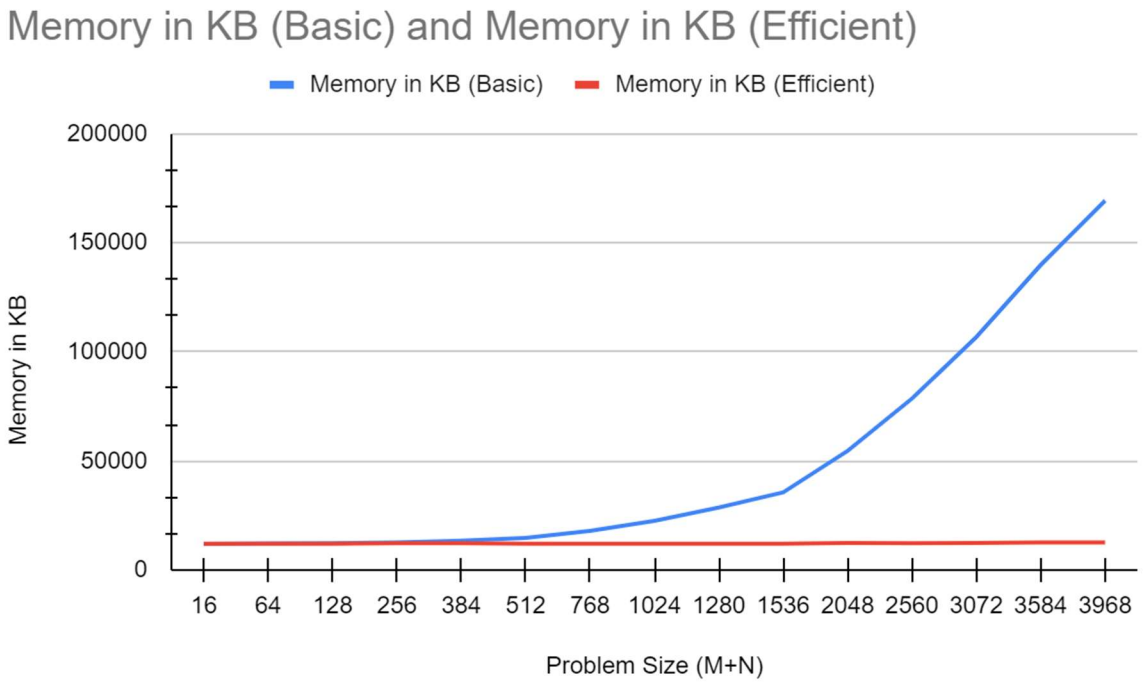
USC ID/s: 2850400993

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	3.032684326171875	9.300708770751953	11880	12044
64	3.4770965576171875	9.982585906982422	12140	12024
128	6.953001022338867	13.849973678588867	12232	12072
256	18.570899963378906	31.17966651916504	12624	12260
384	31.673908233642578	61.95878982543945	13480	12300
512	57.96074867248535	91.77446365356445	14664	12032
768	128.7863254547119	214.37621116638184	17872	12028
1024	220.0784683227539	371.7331886291504	22476	12060
1280	347.4297523498535	521.1272239685059	28652	12028
1536	478.0464172363281	728.8472652435303	35588	12092
2048	847.9204177856445	1292.328119277954	54552	12372
2560	1358.4742546081543	2063.9243125915527	78528	12324
3072	1964.1072750091553	2911.9534492492676	106724	12408
3584	2704.451322555542	3933.2165718078613	139836	12632
3968	3482.0382595062256	4945.048093795776	169292	12636

Datapoints

Insights

Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Exponential

Efficient: Linear

Explanation:

Let m and n be string length of string 1 and string 2 respectively.

In the basic version of sequence alignment, the memory requirement is O(M*N) as it creates a memoization table of size M*N therefore it has exponential space complexity.

whereas the efficient version requires at most space of O(M) or O(N) since at each divide and conquer step it requires at most 2*M or 2*N memoization table which can be reused again so, its space complexity is linear.

Graph2 – Time vs Problem Size (M+N)

Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Exponential

Efficient: Exponential

Explanation:

Let M and N be string length of string 1 and string 2 respectively. Let C be the constant time for computation.

In the basic version of sequence alignment, the total time requirement is O(C*M*N) where C is the computations required to find the values of each element in the DP array.

So, the time complexity of basic version ≈ O(M*N)

For the efficient version work done at each step reduces to half of the previous step (as the subproblem is dividing the problem into halves). Here the work is done to find the values of the DP array.

work done = C*M*N + C*M*N/2 + C*M*N/4 + ... = O(2*C*M*N)

Therefore, the time complexity of efficient version ≈ 2*O(M*N) ≈ 2 * time complexity of basic version.

Hence time complexity of both basic as well as efficient versions is exponential in nature but the time complexity of the efficient version increases at twice the rate of the basic version.

Contribution

(Please mention what each member did if you think everyone in the group does not have an equal contribution, otherwise, write “Equal Contribution”)

<USC ID/s>: <Equal Contribution>