15210: Parallel and Sequential Data Structures and Algorithms

DPLab

Zikang Wang (zikangw)

4.1

$S[i,j]$: substring of S from i to j, inclusive

Recursive Solution:

$DP(i)$: represents whether or not S[i,n] can be split into valid words.

$$\begin{cases} DP(i) = \bigvee_{j=i+i}^{n} isWord(S[i,j]) \wedge DP(j) \\ DP(n) = true \end{cases}$$

Go from DP(n) to DP(0), DP(0) is the final answer.

Sharing:

There are n^2 distinct calls (between any of the two letters) for isWord(), and n+1 call of the DP function

DAG and Cost:

We have $O(n^2)$ nodes in the DAG. Each represents a substring from i to j.

For each sub-problem, the work is O(n) since there are at most n words to check. And we have n+1 problems, 0~n. Therefore the work is $O(n^2)$

4.2

Transform:

Label villages on the left side of the river 0 ~n-1 and assume their locations are following the ascending order.

Label corresponding villages on the right side of the river 0 ~ n-1, too, but it can be in any order. Represent the order of these villages with an array S. S is a permutation of 0~n-1.

To avoid crossing, the order of S should be also in the ascending order. Therefore, the problem is to find a longest increasing subsequence of S.

Recursive Solution:

DP(i): length of longest increasing subsequence that ends with S(i)

$$\begin{cases} DP(i) = 1 + max_{j<i, A[j]<A[i]} DP(j) \\ DP(0) = 1 \end{cases}$$

DP(n) is the solution

Sharing:

We need to calculate n states, one for each position. And for each position i, we need to do i-1 comparisons.

DAG and Cost:

There are n nodes in the DAG, each represents the length of longest increasing sequence that ends on this element.

For each position, we need to do at most n-1 comparisons. There are n positions. Therefore the cost is $O(n^2)$

4.3

4.4

Recursive Solution:

$R[s]$: rules, $1 \le s \le k$

$S[i,j]$: substring from $S[i]$ to $S[j]$, inclusive

$DP[s][i][j]$: min number of steps needed to go from σ_s to $S[i,j]$. $DP[s][i][j] = \infty$ if it is not reachable.

$$\begin{cases} DP[s][i][j] = 1 + min_{(u,v)\in R}\{min_{1\le w<j}\{max\{DP[u][i][w], DP[v][w+1][j]\}\}\} \\ DP[s][i][i] = 0 \; if \; reachable \\ DP[s][i][i] = \infty \; if \; unreachable \end{cases}$$

Final solution is $DP[1][0][n-1]$

Sharing:

There are kn^2 DP sub-problems. For each one we need to go through at most m rules and n characters.

DAG and Cost:

Each node represents the steps from one character to a substring of S; there are n^2 substrings and k characters therefore kn^2 nodes. For each node we have $O(nm)$

work. Total work is $O(kn^3m)$.