Class 11 :(:Dynamic :programming :)

Tuesday, September 28, 2021 6:26 AM

Previous classes:

we studied the following dynamic programming problems in the previous classes.

i) Shortest paths in DAGs
ii) Longest increasing subsequence
iii) Edit distance.

We see the following dynamic problems in this lecture.
i) chain matrix multiplication
ii) Knapsack

Chain matrix multiplication

well the name itself must be suggesting that we have a chain of matrices to multiply.
We have already went through matrix multiplication in divide and conquer where the upper bound using naive algorithm is O(mnp) where mxn and nxp are two matrices considered in multiplication.

Whats the problem here ??

we have n matrices, and we select a pair of matrices for multiplication in each step of simplification.
The problem here is how we have to select the two matrices so that the total multiplication is optimal.

For dynamic programming we need subproblems.

In this case what might be the subproblems ???

Every step has two matrices to multiply.
So a sub problem requires that the matrix multiplication at that step is optimal.

Lets have some illustrations which give more intuition into this problem.

Lets say we have A_1, A_2, A_3, ..., A_n matrices to multiply in the same order.
i.e., we have to find $A_1 \times A_2 \times A_3 \times .. .. \times A_n$
assuming that the number of columns in A_i is same as A_{i+1} .
when we are multiplying these, we don't just multiply all at a time. We divide it into pairs of subproblems..

Method 1:
Pseudo code:

```
M(A,n):
    let R = A1
    for i=2 to n
        R <-- R*Ai
    return R
```

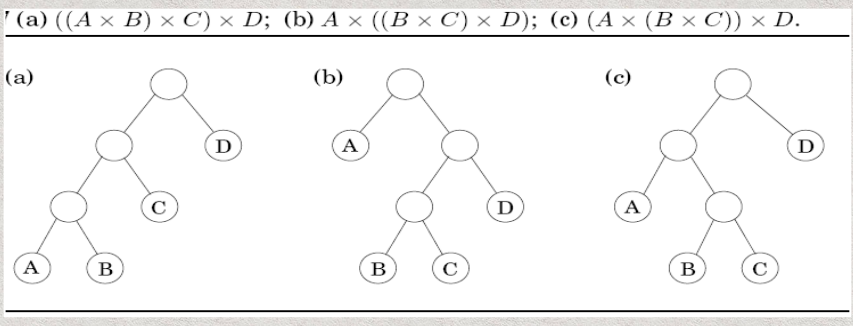
Method 2:
Pseudo code:

```
call M(A,0,n-1);
M(A,low,high):
    mid <- (low+high)/2
    Return M(A,low,mid)*M(A,mid+1,high)
```

Method 1 is naive ..
Method 2 is a bit dynamic and it is like mergesort. We have quick sort as superset of mergesort. So we can use something like quick sort that can make this multiplication more optimal.

Let analyze this in a pictorial mannar.
what kind of pictures do we have in algorothms?
GRAPHS!!!

Every node is a matrix and we join two nodes to a node where get its product.



We want the multiplication at a node to be optimal implies the product at their children is optimal.

Its time to get back to our math analysis in this problem.

We have $A_1 \times A_2 \times A_3 \times .. .. \times A_n$
let $A_i \times A_{i+1} \times .. .. \times A_j$ be the subproblem we solve before solving the actual one.

Let complexity of doing $A_i \times A_{i+1} \times .. .. \times A_j$ be C(i, j)
how to define C(i ,j)???
$A_i \times A_{i+1} \times .. .. \times A_j$ can be divided into two subproblems having
$A_i \times A_{i+1} \times .. .. \times A_k$ and $A_{k+1} \times A_{k+1} \times .. .. \times A_j$
so complexity
$C(i, j) = \min\{ C(i,k) + C(k+1,j) + r(i)*c(k)*c(j) \}$ for all $i \leq k < j$
where r(m), c(m) is number of rows, columns in matrix A_m

We will have a 2D matrix C of order n X n
with C(i,j) as entry at i,j position.
Each entry takes O(n) time since it has to go through i to j
Hence overall complexity is $O(n)*n^2 = O(n^3)$

Here we calculated only the complexity and path to optimal solution but not the optimal solution.

Knapsack

Once upon a time there was a theif. He has a knapsack to carry his booty. One night he enters a house of an old man and demands old man to give the location of valuables in the house. The old man was very busy writing something on a paper. So he just showed the theif the shelf infront of him and said, "That's all the valuables I have!"
The theif went there and saw that each item had a paper beside it with its value written on it. He wanted to carry all of them but, his knapsack can carry only maximum of W weight.
He wanted to select only as many as the most valuable ones.

Is he correct at his descision ??
No..

He found that he could carry 3 small things whose total weighes less than the most valuable one and also total value is more than that of most valuable one.

Poor theif was disappointed at his greed and his stupidity of choosing most valuable ones.

He looked at the old man who was writing something on paper.
The theif asked him for a paper and a pen which the busy old man gave him without any resistance.

The theif wrote down the values and weights of items on it and he tried to find those items that add upto a weight of less than or equal to what his knapsack can carry and total value adding to maximum.

He used lot of trial and error ..
Meanwhile , the old man notices the theif doing something ans asks him what he was trying to solve.
Knowing the problem of theif , the old man comes up with the following explaination.

Let
w_1 , w_2 , w_3 .. , w_n be weights of items
v_1 , v_2 , v_3 .. , v_n be the corresponding values of the items.

And let the Weight your knapsack can carry be W.
For every item either you include it or do not include it.
Lets say K(w,i) be the total value of items selected from 0 to i items

call K(w,n)
K(w, i):
    if(i=0) then return 0;
    # include item i and do not inlcude it and find max of both
    if(w_i>w) # then we donot include it
        K(w,i) = K(w,i−1)
    else
        K(w,i) = max (K(w,i−1) , K(w−w_i , i−1)+ V_i)
    return K(w,i)
The complexity will be O(n)*W = O(nW)

since O(n) for total number of items and W for the max depth of the recurrsion.
Thus the theif was very happy at the old man's idea and visited him everyday to learn how to solve such problems..

Guess who the old man is ..
A professor who teaches algorithms.
He may not have added more into this explaination in the interest of time as he had another important work to do.

So .. I end here as I have other assignments to do.

– Miryala Narayana Reddy
2020101044
ug2k20