More Dynamic Programming

Matrix Chain Multiplication

Matrix Chain Multiplication

- Given some matrices to multiply, determine the ***best*** order to multiply them so you minimize the total number of single element multiplications.
  - i.e. Determine the way the matrices are parenthesized.

- First off, it should be noted that matrix multiplication is associative, but not commutative. But since it is associative, we always have:

- ((AB)(CD)) = (A(B(CD))), or any other grouping as long as the matrices are in the same consecutive order.

- BUT NOT: ((AB)(CD)) = ((BA)(DC))

Matrix Chain Multiplication

- It may appear that the amount of work done won't change if you change the parenthesization of the expression, but we can prove that is not the case!

- Let us use the following example:
  - Let A be a 2x10 matrix
  - Let B be a 10x50 matrix
  - Let C be a 50x20 matrix
- Try (AB)C vs A(BC)
- But FIRST, let's review some matrix multiplication rules…

Multiplyting two Matrices (C = AB)

- Dimension of matrix A = n * m
- Dimension of matrix B = m * l
- Dimension of matrix C = n * l (how ?)
- Cost of multiplying these two matrices = n * m * l , how ?

A = [a_{00} a_{01} a_{02}
a_{10} a_{11} a_{12}]
a_{20} a_{21} a_{22}

B = [b_{00} b_{01} b_{02} b_{03}
b_{10} b_{11} b_{12} b_{13}
b_{20} b_{21} b_{22} b_{23}]

Lets see how many calculations do we make for C

Matrix Chain Multiplication

- Let's get back to our example: We will show that the way we group matrices when multiplying A, B, C ***matters***:
  - Let A be a 2x10 matrix
  - Let B be a 10x50 matrix
  - Let C be a 50x20 matrix

- Consider computing **A(BC):**
  - # multiplications for (BC) = 10x50x20 = 10000,
  - D = BC, dimension of D = 10 * 20
  - # multiplications for A(D) = 2x10x20 = 400
  - Total multiplications = 10000 + 400 = 10400.

- Consider computing **(AB)C**:
  - # multiplications for (AB) = 2x10x50 = 1000,
  - E = (AB), dimension of E = 2 x 50
  - # multiplications for (E)C = 2x50x20 = 2000,
  - Total multiplications = 1000 + 2000 = 3000

Matrix Chain Multiplication

- Thus, our **goal** today is:
- Given a chain of matrices to multiply, determine the fewest number of multiplications necessary to compute the product, also calculate the final answer matrix

Matrix Chain Multiplication

- The key to solving this problem is noticing the ***sub-problem optimality condition***:
  - If a particular parenthesization of the whole product is optimal, then any sub-parenthesization in that product is optimal as well.
- ***Say What?***
  - ***If*** (A (B ((CD) (EF)))) is optimal
  - Then (B ((CD) (EF))) is optimal as well
  - ***Proof on the next slide…***

Matrix Chain Multiplication

- Assume that we are calculating ABCDEF and that the following parenthesization is optimal:
        - (A (B ((CD) (EF))))
    - Then it is necessarily the case that
        - (B ((CD) (EF)))
    - is the optimal parenthesization of BCDEF.

- Why is this?
    - Because if it wasn't, and say (((BC) (DE)) F) was better, then it would also follow that
        - (A (((BC) (DE)) F)) was better than
        - (A (B ((CD) (EF)))),
    - contradicting its optimality!

Matrix Chain Multiplication

- Our final multiplication will ALWAYS be of the form
  - $(A_0 \cdot A_1 \cdot \ldots A_k) \cdot (A_{k+1} \cdot A_{k+2} \cdot \ldots A_{n-1})$
- In essence, there is exactly one value of k for which we should "split" our work into two separate cases so that we get an optimal result.
  - Here is a list of the cases to choose from:
  - $(A_0) \cdot (A_1 \cdot A_{k+2} \cdot \ldots A_{n-1})$
  - $(A_0 \cdot A_1) \cdot (A_2 \cdot A_{k+2} \cdot \ldots A_{n-1})$
  - $(A_0 \cdot A_1 \cdot A_2) \cdot (A_3 \cdot A_{k+2} \cdot \ldots A_{n-1})$
  - ...
  - $(A_0 \cdot A_1 \cdot \ldots A_{n-3}) \cdot (A_{n-2} \cdot A_{n-1})$
  - $(A_0 \cdot A_1 \cdot \ldots A_{n-2}) \cdot (A_{n-1})$
- Basically, count the number of multiplications in each of these choices and **pick the minimum**.
  - One other point to notice is that you have to account for the minimum number of multiplications in each of the two products.

Matrix Chain Multiplication

- Consider the case multiplying these 4 matrices:
  - A: 2x4
  - B: 4x2
  - C: 2x3
  - D: 3x1

- 1. (A)(BCD) - This is a 2x4 multiplied by a 4x1,
  - so 2x4x1 = 8 multiplications, plus whatever work it will take to multiply (BCD).

- 2. (AB)(CD) - This is a 2x2 multiplied by a 2x1,
  - so 2x2x1 = 4 multiplications, plus whatever work it will take to multiply (AB) and (CD).

Matrix Chain Multiplication

- **Our recursive formula:**
  - $cost(i, j) = cost(i, k) + cost(k + 1, j) + p_{i-1} \cdot p_k \cdot p_j$
- Let's solve it using recursion

```cpp
#define ordered_set tree<ll, null_type,less<ll>, rb_tree_tag,tree_order_statistics_no
int dp[105][105];

int MCM(vector<int>&arr,int x,int y)
{
        if(x==y)
                return 0;
        if(dp[x][y]!=-1)
                return dp[x][y];
        int ans=INT_MAX;
        for(int i=x;i<y;i++)
                ans=min(ans,MCM(arr,x,i)+MCM(arr,i+1,y)+arr[i]*arr[x-1]*arr[y]);
        return dp[x][y]=ans;
}

int main()
{
        boost
        int t=1;
        cin>>t;
        while(t--)
        {
                int n;
                cin>>n;
                memset(dp,-1,sizeof(dp));
                vector<int>arr(n);
                for(int i=0;i<n;i++)
                        cin>>arr[i];
                cout<<MCM(arr,1,n-1)<<"\n";
        }
        return 0;
}
```

Download as text