Matrix Chain Multiplication

এখন ধরা যাক আমাদেরকে ৩টা ম্যাট্রিক্সের গুণফল $A_1 \times A_2 \times A_3$ বের করতে হবে। এখন $col_1 = row_2$ এবং $col_2 = row_3$ হলেই শুধুমাত্র আমরা গুণটা করতে পারবো। গুণটা দুই উপায়ে করা যায়:

- $(A_1 \times A_2) \times A_3$
- $A_1 \times (A_2 \times A_3)$

- $(A_1 \times A_2) \times A_3$
- $A_1 \times (A_2 \times A_3)$

দুইটা উপায়ের পার্থক্য শুধু ব্র্যাকেটিং এ। দুই ক্ষেত্রেই আমরা $row_1 \times col_3$ ডাইমেনশনের একটা ম্যাট্রিক্স পাবো। কিন্তু দুইটা উপায়েই কি আমাদের একই সংখ্যক স্কেলার গুণ করা লাগবে? একটা উদাহরণ দেখি।

মনে করো ম্যাট্রিক্সগুলোর ডাইমেনশন হলো যথাক্রমে (10×100), (100×5) এবং (5×50)

- $(A_1 \times A_2) \times A_3$ ব্র্যাকেটিং এ স্কেলার গুণ করতে হবে $(10 \times 100 \times 5) + (10 \times 5 \times 50) = 7500$ বার।
- $A_1 \times (A_2 \times A_3)$ ব্র্যাকেটিং এ স্কেলার গুণ করতে হবে $(100 \times 5 \times 50) + (10 \times 100 \times 50) = 75000$ বার।

যেমন $n=4$ এর জন্য $f(0,3)$ কে এভাবে ভাগ করা যায়:

- $k=0$ ব্রাকেটিং -> $(A_0)\times(A_1\times A_2\times A_3)$ সাবপ্রবলেম -> $f(0,0)+f(1,3)$
- $k=1$ ব্রাকেটিং -> $(A_0\times A_1)\times(A_2\times A_3)$ সাবপ্রবলেম -> $f(0,1)+f(2,3)$
- $k=2$ ব্রাকেটিং -> $(A_0\times A_1\times A_2)\times(A_3)$ সাবপ্রবলেম -> $f(0,2)+f(3,3)$

অর্থাৎ আমরা যতভাবে সম্ভব অ্যারেটাকে দুইভাগে ভাগ করে ফেলবো এবং সাবপ্রবলেমগুলো রিকার্সিভলি সলভ করবো, প্রতিটা k এর জন্য সাবপ্রবলেম হবে $f(i,k)$ এবং $f(k+1,n-1)$।

ম্যাট্রিক্সগুলো দুইভাগ করেই কিন্তু কাজ শেষ না, এবার মার্জ করতে হবে। k তম পজিশনে ভাগ করলে তুমি বামে $row_i \times col_k$ সাইজের এবং ডানে $row_{k+1} \times col_j$ সাইজের ম্যাট্রিক্স পাবে যেখানে $col_k = row_{k+1}$। এবার এই দুইটি ম্যাট্রিক্সও গুণ করতে হবে এবং অপারেশন লাগবে $row_i \times col_k \times col_j$ টা।

ম্যাট্রিক্সগুলো দুইভাগ করেই কিন্তু কাজ শেষ না, এবার মার্জ করতে হবে। k তম পজিশনে ভাগ করলে তুমি বামে $row_i \times col_k$ সাইজের এবং ডানে $row_{k+1} \times col_j$ সাইজের ম্যাট্রিক্স পাবে যেখানে $col_k = row_{k+1}$। এবার এই দুইটি ম্যাট্রিক্সও গুণ করতে হবে এবং অপারেশন লাগবে $row_i \times col_k \times col_j$ টা।

ডিভাইড এন্ড কনকোয়ারে ২টা মুল কাজ থাকে, ডান আর বামের সাবপ্রবলেম ডিফাইন করা এবং মার্জ করা। আমরা মার্জ অপারেশনটাকে আলাদা ফাংশন হিসাবে ধরলে আরো পরিষ্কার একটা ফর্মুলা লিখতে পারি:

$$mergeCost(i, j, k) = mat[i].row * mat[k].col + mat[j].col)$$
$$f(i, i) = 0$$
$$f(i, j) = min(f(i, k) + f(k + 1, j) + mergeCost(i, j, k) \ where \ k \in \ [i, j - 1]$$

```
int mergeCost(int i, int j, int k) {
    return mats[i].row * mats[k].col * mats[j].col;
}

int f(int i, int j) {
    if (i >= j) {
        return 0;
    }

    if (mem[i][j] != EMPTY_VALUE) {
        return mem[i][j];
    }

    int ans = INF;
    for(int k = i;k < j;k++) {
        int res_left = f(i, k);
        int res_right = f(k + 1, j);
        int cost = (res_left + res_right) + mergeCost(i, j, k);
        ans = min(ans, cost);
    }

    mem[i][j] = ans;
    return mem[i][j];
}
```