Matrix Chain Multiplication

([A1((A2 A3) A4)) A5)

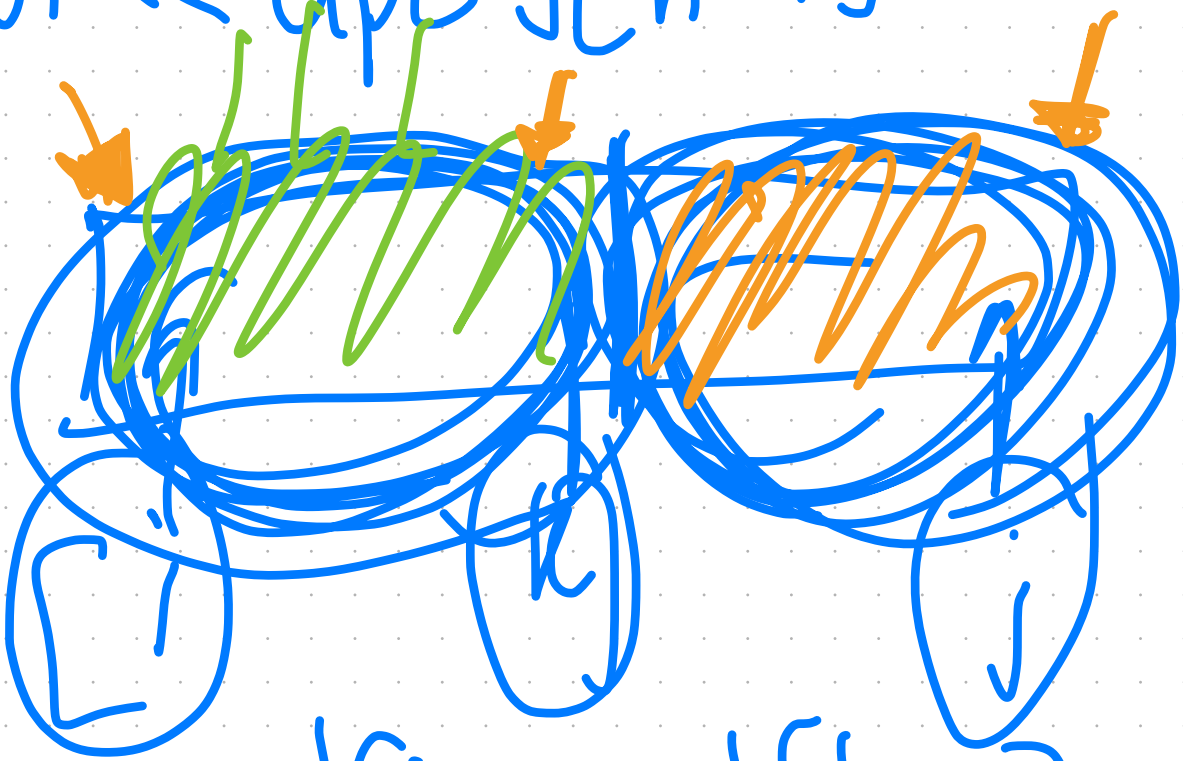
→ Top-down ✓✓✓

→ Bottom-up ***

```
int row[20], col[20], dp[20][20], i,
  path[20][20], n;

main()
  cin >> n
  for(i=0 to n)
    cin >> row[i] >> col[i]
  for(m=1; m<n; m++)        ขนาดของ
    for(i=0; i<n-m; i++)    จุดเริ่ม
    {  j = i+m;             j คือจุดจบ
       dp[i][j] = ∞
```

```
for(k=i; k<j; k++)   k คือตัวแบ่ง
{
  now = dp[i][k] + dp[k+1][j]
  + row[i] * col[k] * col[j];
  if(now < dp[i][j])
  {
    dp[i][j] = now
    path[i][j] = k
  }
}
cout << dp[0][n-1]
```



col[k] = col[k+1]

Print(0, n-1)

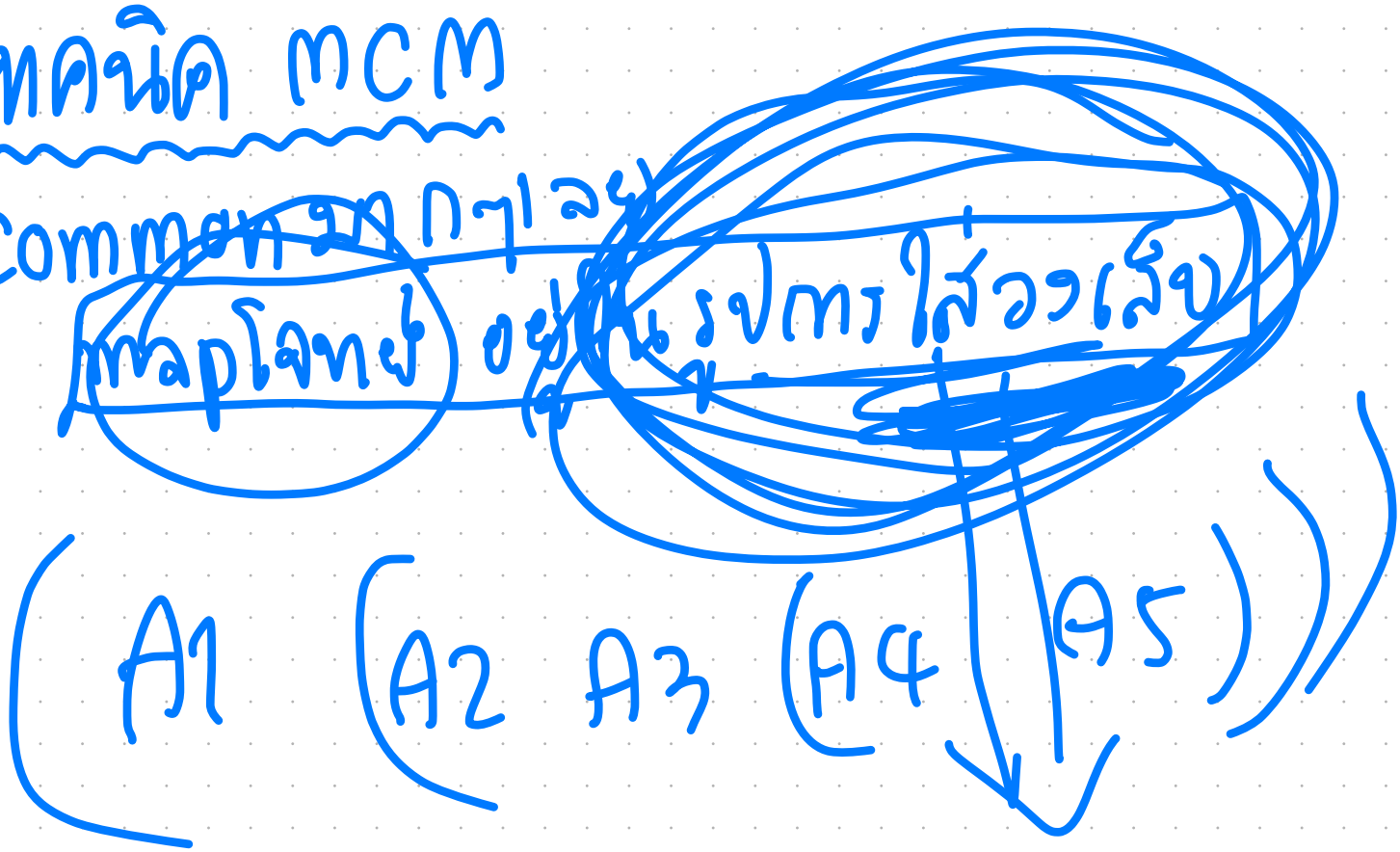
(5×10) (10×20)

```
void P(int l, int r)
  if(l==r)
     printf("A%d", l+1);
  else
  {
     printf("(");
      P(l, path[l][r]);
     printf(" x ");
     P(path[l][r]+1, r);
     printf(")");
  }
}
```

เทคนิค MCM

Comment ตกลง

map โจทย์ อยู่ในรูปการใส่วงเล็บ



ข้อ MCM ลบ.

→ Seq ไม่เกิน 500

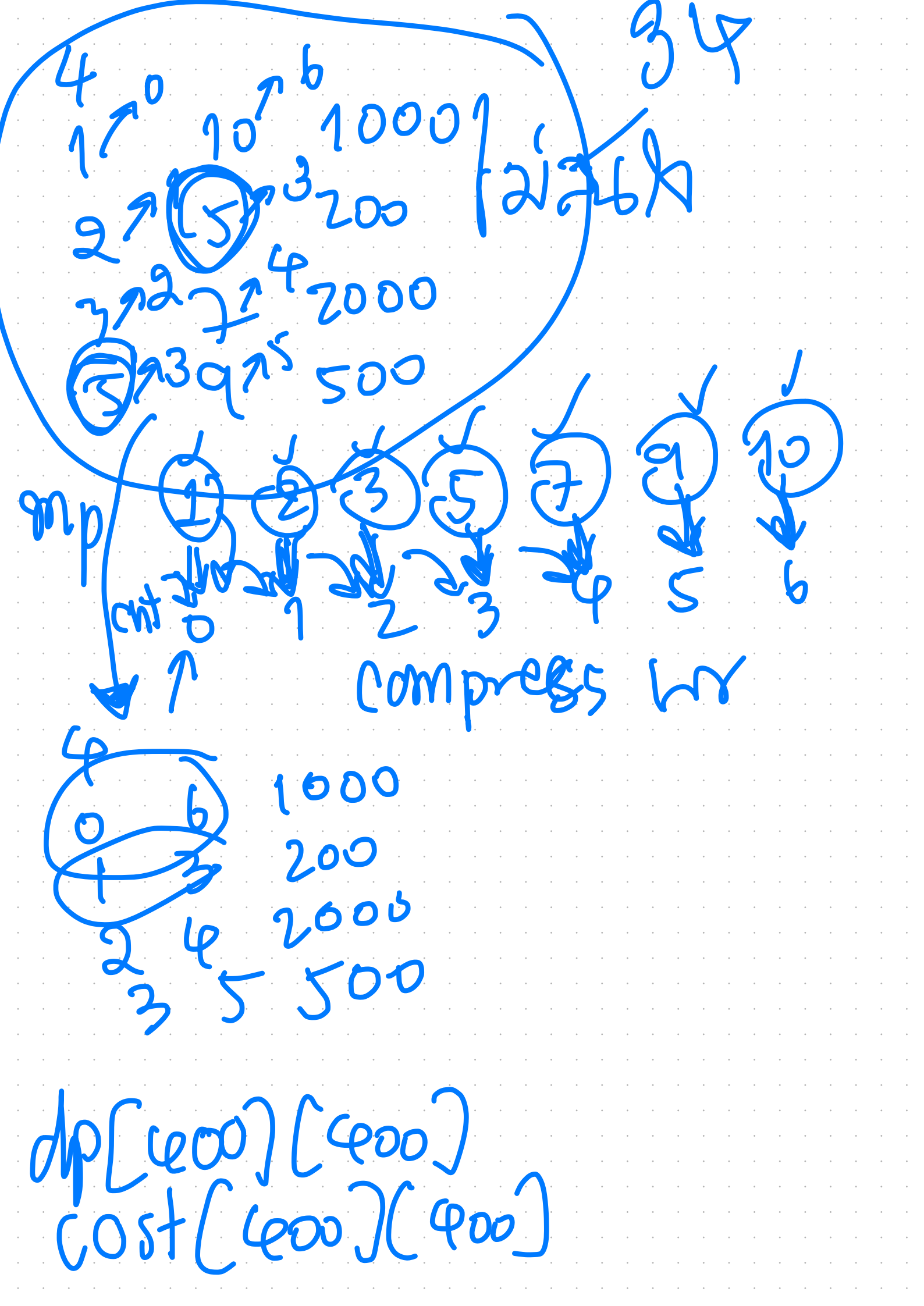
dp[500][500] / cost[500][500]

```
unordered_map<int, int> mp;
cin >> n
for i to n
    cin >> l[i] >> r[i] >> c[i]
if(mp[ l[i] ] != 1) ← เช็คไม่ให้ซ้ำ
{   mp[ l[i] ] = 1; ←
```

```
        g.push_back(l[i]);
    }
    if(mp[r[i]]!=1)
    {
        mp[r[i]]=1;
        g.push_back(r[i]);
    }
}
sort(g.begin(),g.end());
cnt=0;
for(auto x:g)
    mp[x]=cnt++;
for(i=0;i<n;i++)
    cost[mp[l[i]]][mp[r[i]]]
        =c[i];
```

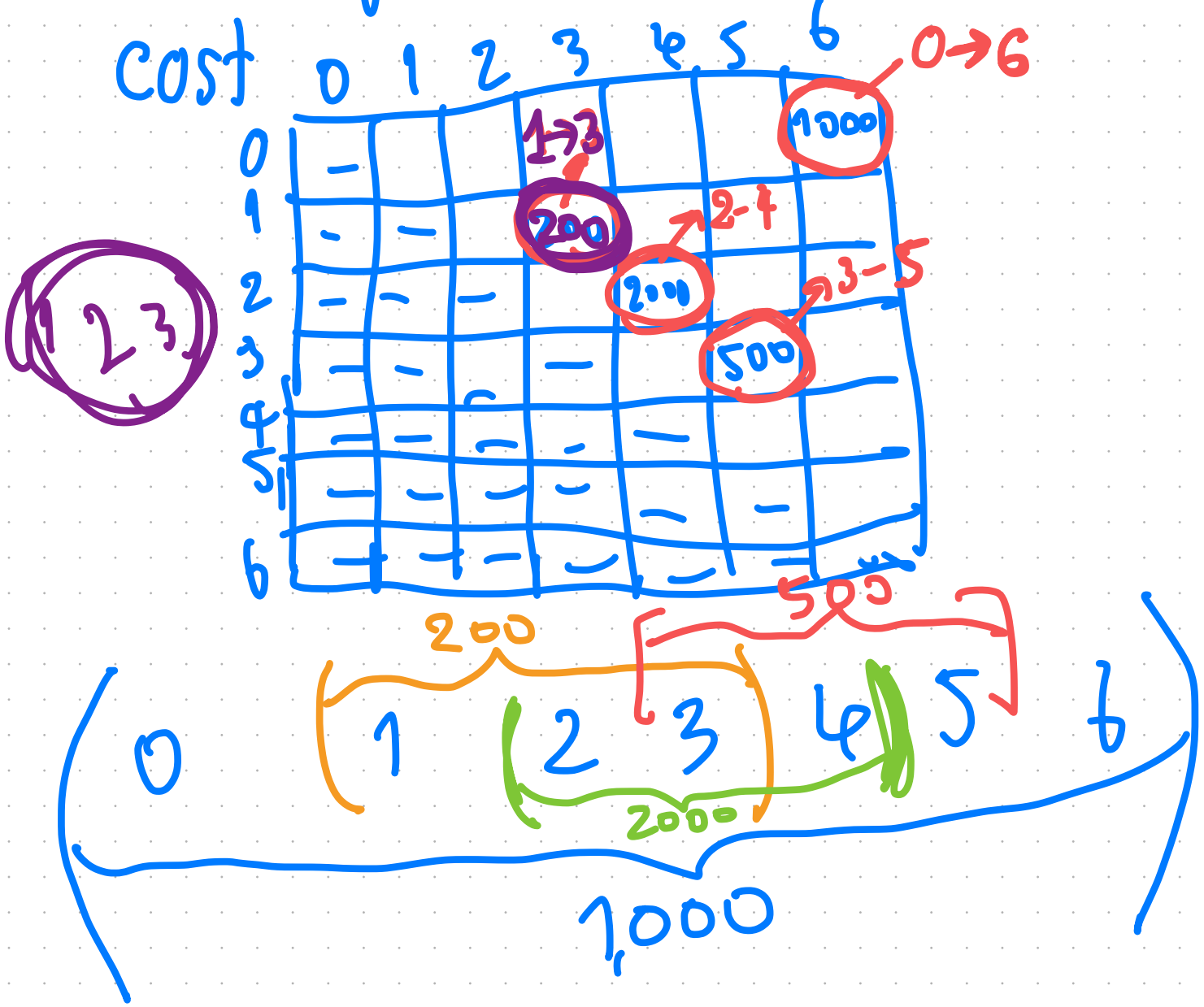
12

23


4
1 → 0
10 → 6
1000
2 → 5 → 3
200
3 → 2 → 7 → 4
2000
5 → 3 → 9 → 5
500
34
mp
1
2
3
5
7
9
10
cnt
0
1
2
3
4
5
6
compress
4
0
6
1
3
2
4
5
1000
200
2000
500
dp[400][400]
cost[400][400]

dp[i][j] = รายการอาหารช่วง i ถึง j
ทำเงินได้มากสุด กี่บาท

คำตอบอยู่ที่ dp[0][cnt-1]



main ตอน play(0, cnt-1)

return dp[l][r] = cost[l][r]

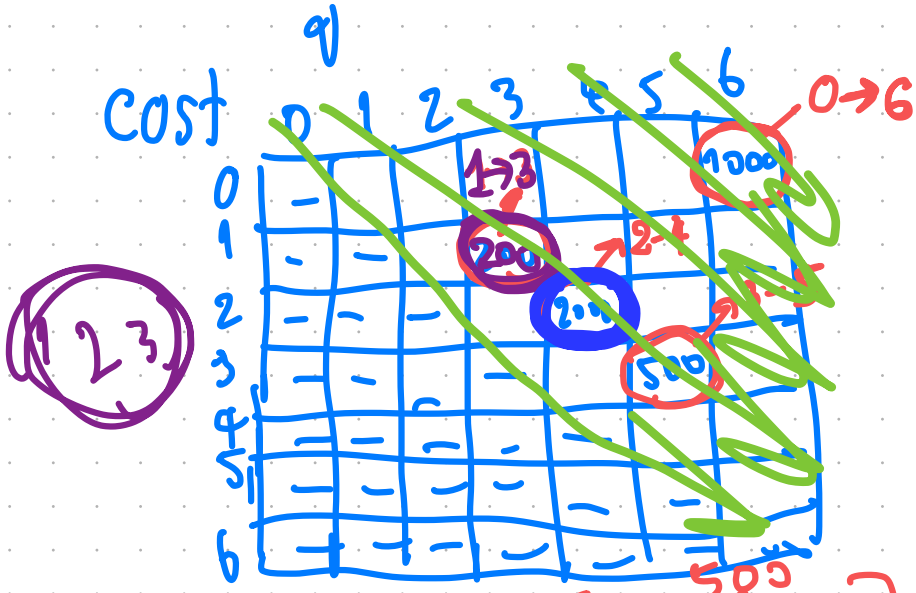
mx = 0

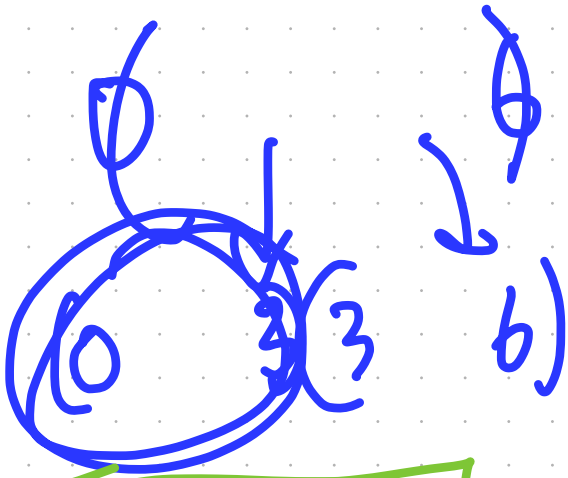
for (k = l + 1; k <= r - 1; k++)

mx = max(mx, play(l, k) + play(k, r));

return dp[l][r] = mx + cost[l][r]

}

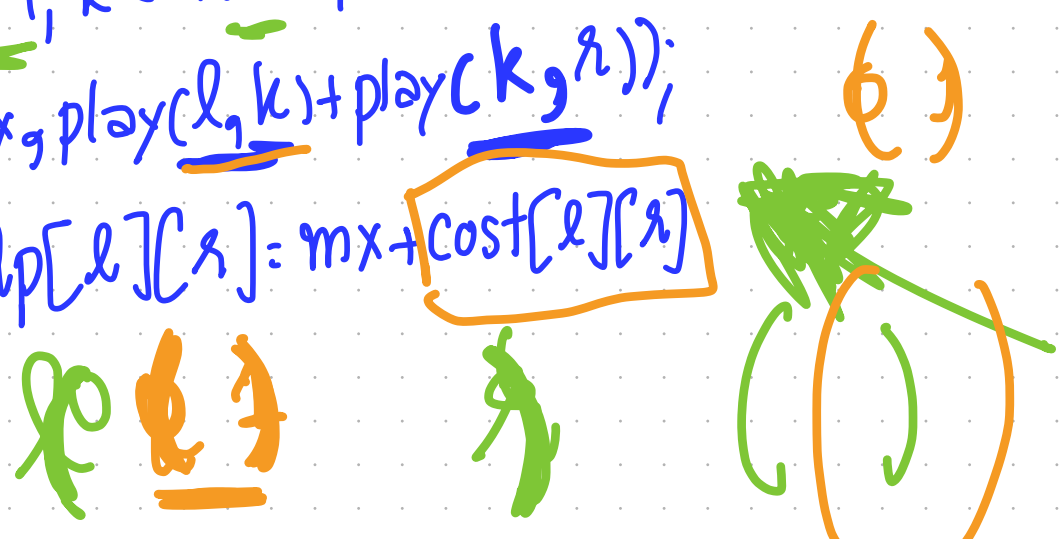


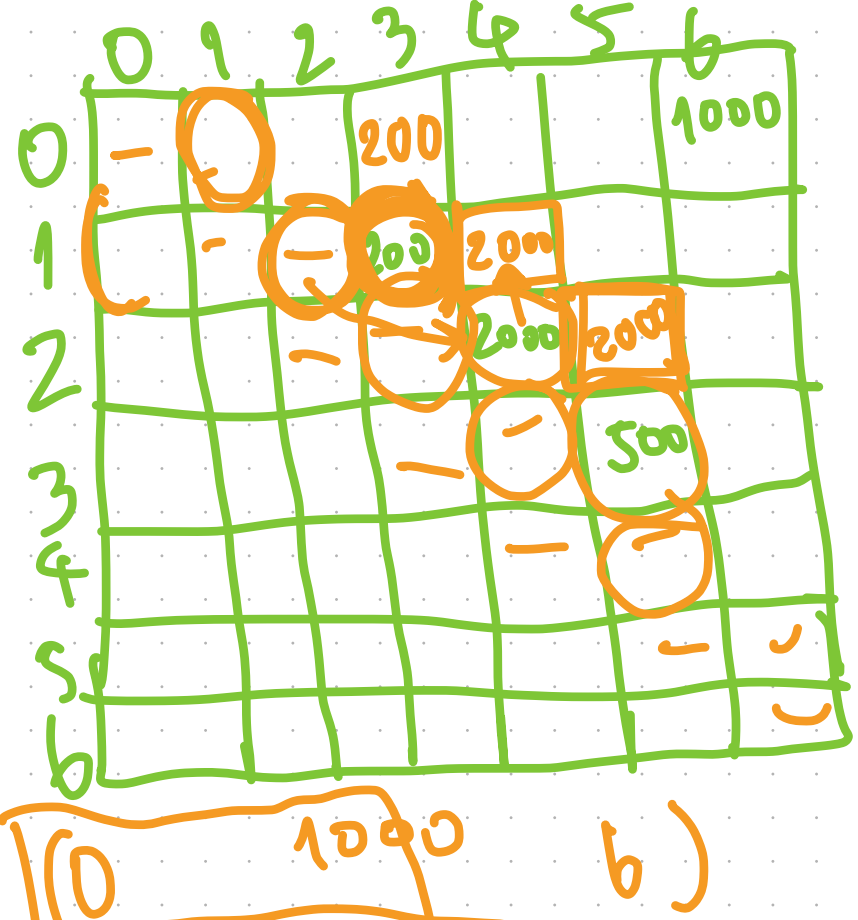


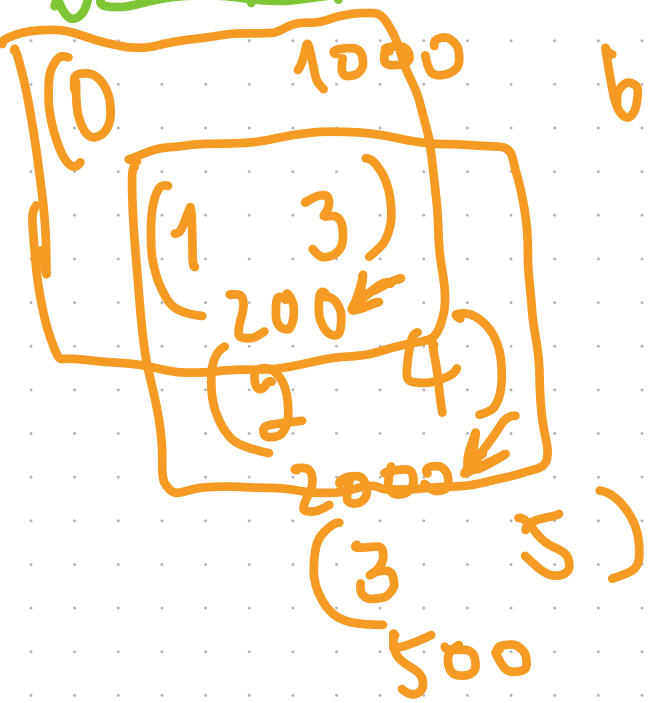
Top down

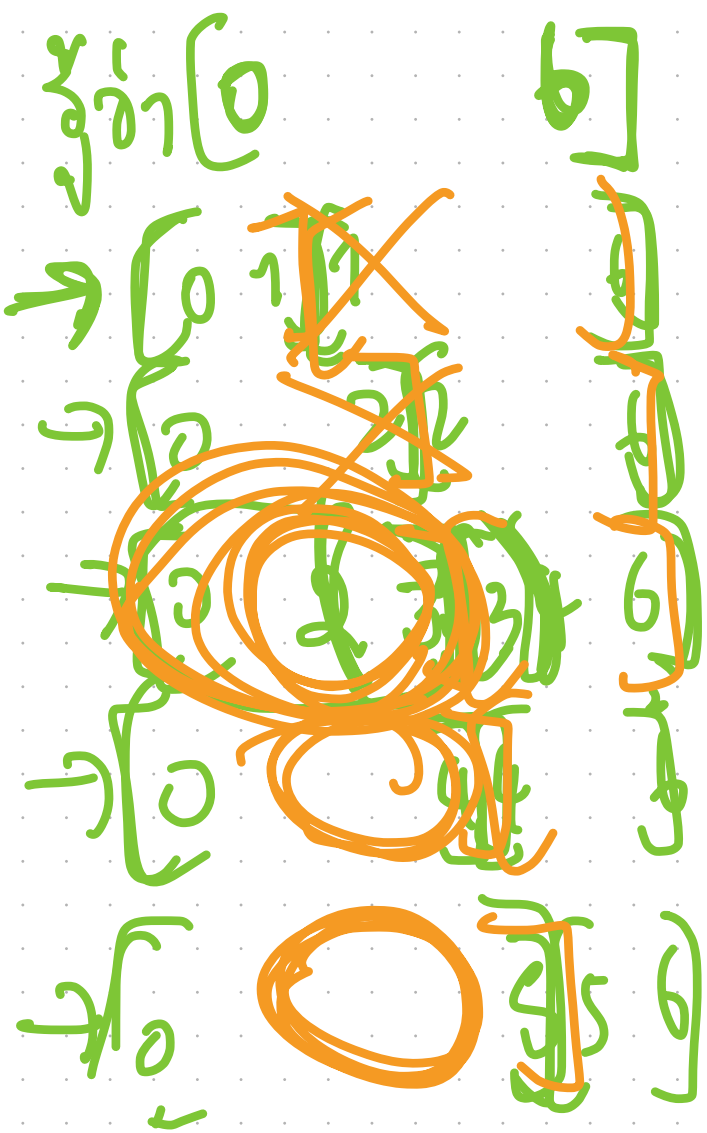
Top down 처럼 cost 1,000 안에 Bottom up

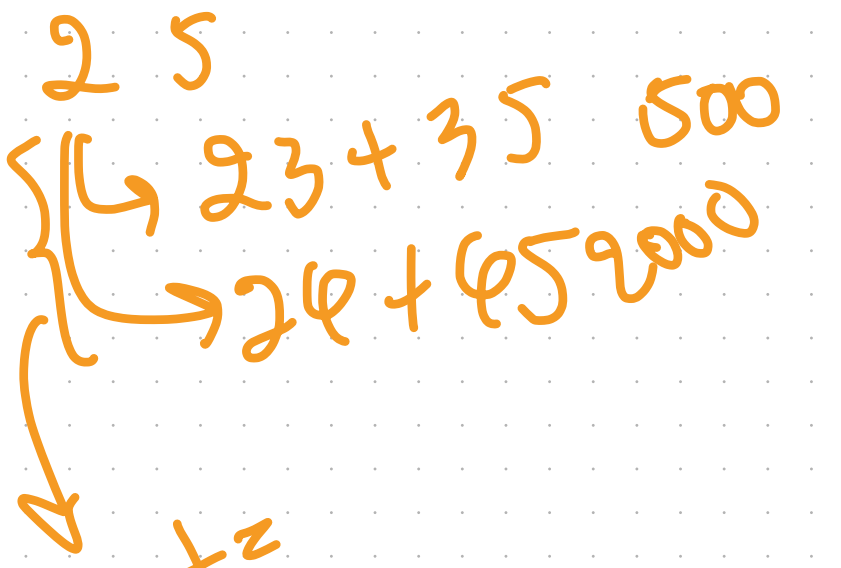

```
play ( l, r )
{
  if( dp[l][r] != -1 )
    return dp[l][r];
  if( l+1 == r )
    return dp[l][r] = cost[l][r]
  mx = 0
  for( k = l+1; k <= r-1 ; k++ )
    mx = max( mx, play(l, k) + play( k, r ));
  return dp[l][r] = mx + cost[l][r]
}
```

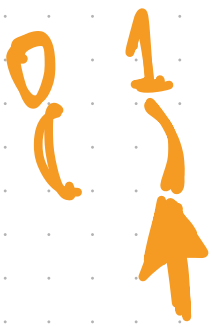



0 1 2 3 4 5 6
1000
200
200
200
200
500


1000
b)
1 3
200
2 4
2000
3 5
500


b


2 5
23+35 500
2e+e5 2000
+z


0 1