第一题

①

dp[i][j] = max( dp[i-1][j] , dp[i-1][j-p[i]] + v[i])

当i = 0时，dp[0][j] = 0，表示没有菜可选，总评价分数为0。

当j = 0时，dp[i][0] = 0，表示报销额度为0，无法点菜，总评价分数为0。

当i > 0 且 j > 0时，考虑两种情况：

一、如果Pi > j，即第i种菜的价格超过了当前报销额度，那么无法选择该菜，因此

dp[i][j] = dp[i-1][j]，表示不选第i种菜，总评价分数不变。

二、 如果Pi <= j，即第i种菜的价格不超过当前报销额度，可以选择该菜。

此时有两种选择：

选择第i种菜：dp[i][j] = dp[i-1][j-Pi] + Vi，表示选择第i种菜，总评价分数为前i-1种菜在报销额度为j-Pi时的最大总评价分数加上第i种菜的评价分数。

不选择第i种菜：dp[i][j] = dp[i-1][j]，表示不选择第i种菜，总评价分数不变。

因此，dp[i][j]的取值为以上两种选择中的最大值。

②

int find\_max\_score(int N, int C, int V[], int P[]) {  
 int dp[MAX\_N + 1][MAX\_C + 1];  
 int i, j;   
 for (i = 0; i <= N; i++) {  
 for (j = 0; j <= C; j++) {  
 dp[i][j] = 0;  
 }  
 }  
 for (i = 1; i <= N; i++) {  
 for (j = 1; j <= C; j++) {  
 if (P[i - 1] > j) {  
 dp[i][j] = dp[i - 1][j];  
 } else {  
 dp[i][j] = (dp[i - 1][j] > (dp[i - 1][j - P[i - 1]] + V[i - 1])) ? dp[i - 1][j] : (dp[i - 1][j - P[i - 1]] + V[i - 1]);  
 }  
 }  
 }  
 return dp[N][C];  
}



第二题

①

dp[i][j] = max( dp[i-1][j] , dp[i-1][j-p[i]] + v[i])

当i = 0时，dp[0][j] = 0，表示没有物品可选，总价值为0。

当j = 0时，dp[i][0] = 0，表示总重量限制为0，无法选择物品，总价值为0。

当i > 0 且 j > 0时，考虑两种情况：

一、 如果wi > j，即第i种物品的重量超过了当前总重量限制，那么无法选择该物品，因此dp[i][j] = dp[i-1][j]，表示不选第i种物品，总价值不变。

二、如果wi <= j，即第i种物品的重量不超过当前总重量限制，可以选择该物品。

此时有两种选择：

选择第i种物品：dp[i][j] = dp[i][j-wi] + vi，表示选择第i种物品，总价值为在剩余重量为j-wi时的最大总价值加上第i种物品的价值。

不选择第i种物品：dp[i][j] = dp[i-1][j]，表示不选择第i种物品，总价值不变。

因此，dp[i][j]的取值为以上两种选择中的最大值。

②

int find\_max\_value(int n, int W, int w[], int v[]) {  
 int dp[MAX\_N + 1][MAX\_W + 1];  
 int i, j;  
 for (i = 0; i <= n; i++) {  
 for (j = 0; j <= W; j++) {  
 dp[i][j] = 0;  
 }  
 }  
  
 for (i = 1; i <= n; i++) {  
 for (j = 1; j <= W; j++) {  
 if (w[i - 1] > j) {  
 dp[i][j] = dp[i - 1][j];  
 } else {  
 int select = dp[i][j - w[i - 1]] + v[i - 1];  
 int N\_select = dp[i - 1][j];  
 dp[i][j] = (select > N\_select) ? select : N\_select;  
 }  
 }  
 }  
 return dp[n][W];  
}

文本

描述已自动生成

第三题

①

dp[i][j] = max( dp[i-1][j] , dp[i-1][j-p[i]] + v[i])

当i = 0时，dp[0][j] = 0，表示没有员工可分配，总盈利为0。

当j = 0时，dp[i][0] = 0，表示没有商店可分配给员工，总盈利为0。

当i > 0 且 j > 0时，考虑两种情况：

一、 如果分配给商店j的员工数超过i，即商店j的员工数大于可分配的员工数，那么无法分配更多的员工给商店j，因此dp[i][j] = dp[i][j-1]，表示不分配员工给商店j，总盈利不变。

二、 如果分配给商店j的员工数不超过i，可以选择分配员工给商店j。

此时有两种选择：

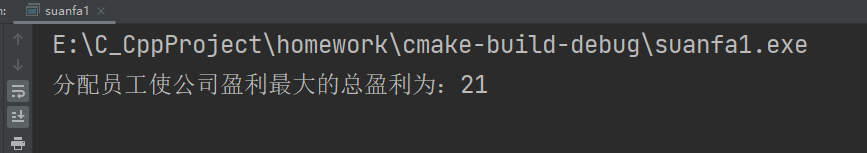
选择分配一个员工给商店j：dp[i][j] = dp[i-1][j-1] + profit[i][j]，表示选择一个员工分配给商店j，总盈利为前一个员工分配给前一个商店时的总盈利加上当前员工分配给当前商店的盈利。

不选择分配员工给商店j：dp[i][j] = dp[i][j-1]，表示不分配员工给商店j，总盈利不变。

因此，dp[i][j]的取值为以上两种选择中的最大值。

②

int max(int a, int b) {  
 return (a > b) ? a : b;  
}  
int find\_max\_profit() {  
 int dp[M+1][N+1];  
 int i, j, k;  
 for (i = 0; i <= M; i++) {  
 for (j = 0; j <= N; j++) {  
 dp[i][j] = 0;  
 }  
 }  
 for (i = 1; i <= M; i++) {  
 for (j = 1; j <= N; j++) {  
 for (k = 0; k <= j; k++) {  
 dp[i][j] = max(dp[i][j], dp[i-1][j-k] + v[i][k]);  
 }  
 }  
 }  
 return dp[M][N];  
}



第四题

①

dp[i][j] = min(dp[i][j], dp[i][k] + dp[k + 1][j] + sum[i][j]) (i <= k <= j - 1)

当i = j时，表示只有一堆石子，无需合并，代价为0，即dp[i][j] = 0。

当i < j时，考虑将第i堆到第j堆石子合并为一堆，可以选择任意一个位置k在[i, j-1]之间进行划分，将问题划分为两部分：

a. 合并第i堆到第k堆石子的代价：dp[i][k]

b. 合并第k+1堆到第j堆石子的代价：dp[k+1][j]

合并后的总代价为石子堆i到k的和加上石子堆k+1到j的和，即dp[i][j] = dp[i][k] + dp[k+1][j] + sum[i][j]，其中sum[i][j]表示第i堆到第j堆石子的总和。

②

int merge\_stones(int stones[], int n) {  
 int dp[MAX\_N][MAX\_N];  
 int sum[MAX\_N][MAX\_N];  
 int i, j, k, l;  
 for (i = 0; i < n; i++) {  
 for (j = 0; j < n; j++) {  
 dp[i][j] = 0;  
 sum[i][j] = 0;  
 }  
 }  
 for (i = 0; i < n; i++) {  
 sum[i][i] = stones[i];  
 for (j = i + 1; j < n; j++) {  
 sum[i][j] = sum[i][j - 1] + stones[j];  
 }  
 }  
 for (l = 2; l <= n; l++) {  
 for (i = 0; i <= n - l; i++) {  
 j = i + l - 1;  
 dp[i][j] = 1e9;   
 for (k = i; k < j; k++) {  
 dp[i][j] = (dp[i][j] < dp[i][k] + dp[k + 1][j] + sum[i][j]) ? dp[i][j] : dp[i][k] + dp[k + 1][j] + sum[i][j];  
 }  
 }  
 }  
 return dp[0][n - 1];  
}

文本

描述已自动生成

第五题

初始化 m[i][j] 和 S[i][j] 的对角线元素，即当 i = j 时，m[i][j] = 0

m[1][1] = 0

m[2][2] = 0

m[3][3] = 0

m[4][4] = 0

遍历矩阵链长度 l，从 2 到 4：

当 l = 2 时：

i = 1, j = 2，k = 1

m[1][2] = m[1][1] + m[2][2] + A[1] × A[2] × A[3] = 0 + 0 + 50 × 10 × 40 = 20000

S[1][2] = 1

i = 2, j = 3，k = 2

m[2][3] = m[2][2] + m[3][3] + A[2] × A[3] × A[4] = 0 + 0 + 10 × 40 × 30 = 12000

S[2][3] = 2

i = 3, j = 4，k = 3

m[3][4] = m[3][3] + m[4][4] + A[3] × A[4] × A[5] = 0 + 0 + 40 × 30 × 5 = 6000

S[3][4] = 3

当 l = 3 时：

i = 1, j = 3，k = 1

m[1][3] = m[1][1] + m[2][3] + A[1] × A[2] × A[4] = 0 + 12000 + 50 × 10 × 30 = 18000

S[1][3] = 2

i = 2, j = 4，k = 2

m[2][4] = m[2][2] + m[3][4] + A[2] × A[3] × A[5] = 0 + 6000 + 10 × 40 × 5 = 8000

S[2][4] = 3

当 l = 4 时：

i = 1, j = 4，k = 1

m[1][4] = m[1][1] + m[2][4] + A[1] × A[2] × A[5] = 0 + 8000 + 50 × 10 × 5 = 8500

S[1][4] = 2

((A1×A2)×A3)×A4