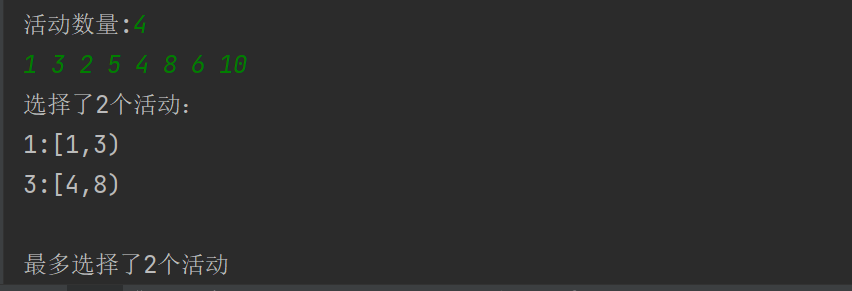
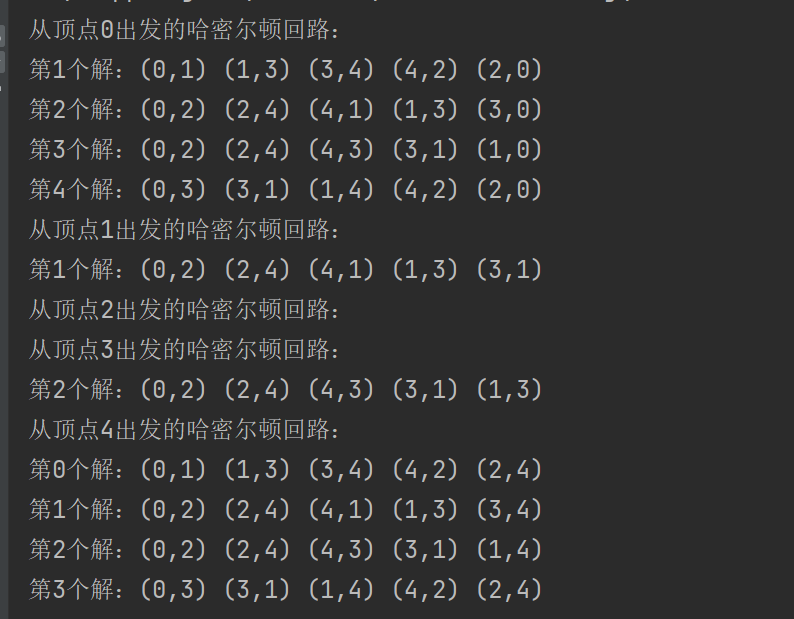
第一题

typedef struct {  
 int id;  
 int start;  
 int end;  
} Activity;   
void backtrack(Activity activities[], int n, int curr, int last\_end, int count, int\* max\_count, int selected[]) {  
 if (curr == n) {  
 if (count > \*max\_count) {\*max\_count = count;  
 printf("选择了%d个活动：\n", count);  
 for (int i = 0; i < n; i++) {  
 if (selected[i]) {  
 printf("%d:[%d,%d)\n", activities[i].id,activities[i].start,activities[i].end);  
 }  
 }  
 printf("\n");  
  
 }  
 return;  
 }  
 if (activities[curr].start >= last\_end) {  
 selected[curr] = 1;  
 backtrack(activities, n, curr + 1, activities[curr].end, count + 1, max\_count, selected);  
 selected[curr] = 0;  
 }  
 backtrack(activities, n, curr + 1, last\_end, count, max\_count, selected);  
}



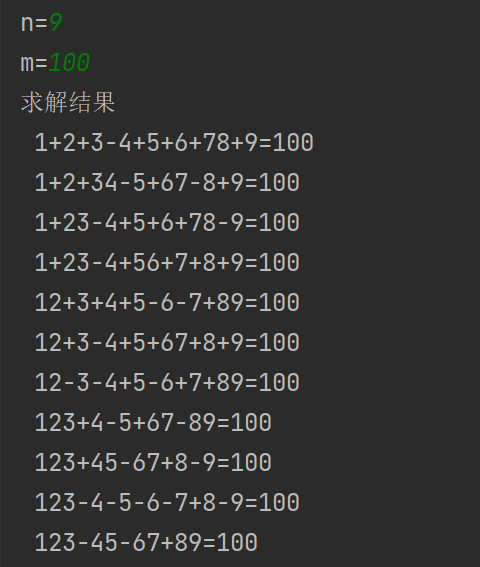
第二题

#define MAXN 20  
typedef struct {  
 int n; // 顶点数  
 bool adj[MAXN][MAXN]; // 邻接矩阵  
} Graph;  
void backtrack(Graph\* g, int curr, int start, int count, int path[]) {  
 if (count == g->n) {  
 if (g->adj[path[count-1]][start]) {  
 printf("找到一个哈密尔顿回路：");  
 for (int i = 0; i < count; i++) {  
 if(i==count-1){  
 printf("(%d,%d)\n", path[i],start);  
 }else{  
 printf("(%d,%d) ", path[i],path[i+1]);  
 }  
 }  
 }  
 return;  
 }  
  
 for (int i = 0; i < g->n; i++) {  
 if (g->adj[path[count-1]][i]) {  
 bool used = false;  
 for (int j = 0; j < count; j++) {  
 if (path[j] == i) {  
 used = true;  
 break;  
 }  
 }  
 if (!used) {  
 path[count] = i;  
 backtrack(g, i, start, count+1, path);  
 path[count] = -1;  
 }  
 }  
 }  
}



第三题

void Count(int M, int N, char op[], int sum, int pre, int a[], int curr) {  
 if (curr == N)   
 {  
 if (sum == M)   
 {  
 printf(" %d", a[0]);  
 for (int j = 1; j < N; j++) {  
 if (op[j] != ' ')  
 printf("%c", op[j]);  
 printf("%d", a[j]);  
 }  
 printf("=100\n");  
 }  
 return;  
 }  
 op[curr] = '+';   
 sum += a[curr];   
 Count(M,N,op, sum, a[curr], a, curr + 1);   
 sum -= a[curr];   
 op[curr] = '-';   
 sum -= a[curr];   
 Count(M,N,op, sum, -a[curr], a, curr + 1);   
 sum += a[curr];   
 op[curr] = ' ';   
 sum -= pre;   
 int temp;  
 if (pre > 0)  
 temp = pre \* 10 + a[curr];  
 else  
 temp = pre \* 10 - a[curr];  
 sum += temp;  
 Count(M,N,op, sum, temp, a, curr + 1);  
}



第四题

#define true 1  
#define false 0  
typedef int bool;  
void backtrack(int curr, bool used[], int nums[], int target) {  
 if (curr == 5) {  
 if (nums[0] \* nums[1] - nums[2] \* nums[3] - nums[4] == target) {  
 printf("%d \* %d - %d \* %d - %d = 1\n", nums[0], nums[1], nums[2], nums[3], nums[4]);  
 }  
 return;  
 }  
 for (int i = 1; i <= 5; i++) {  
 if (!used[i]) {  
 used[i] = true;  
 nums[curr] = i;  
 backtrack(curr + 1, used, nums, target);  
 used[i] = false;  
 }  
 }  
}

文本

描述已自动生成

第五题

int n;  
int c[MAXN+1][MAXN+1];  
int best\_cost = -1;  
int curr\_cost;  
int assign[MAXN+1];  
  
void backtrack(int curr) {  
 if (curr == n) {  
 if (best\_cost == -1 || curr\_cost < best\_cost) {  
 best\_cost = curr\_cost;  
 for (int i = 1; i <= n; i++) {  
 printf("%d ", assign[i]);  
 }  
 printf("Cost: %d\n", best\_cost);  
 }  
 return;  
 }  
  
 for (int i = 1; i <= n; i++) {  
 if (c[i][curr+1] > 0) {  
 curr\_cost += c[i][curr+1];  
 assign[curr+1] = i;  
 c[i][curr+1] = 0;  
 backtrack(curr+1);  
 c[i][curr+1] = curr\_cost - assign[curr+1];  
 curr\_cost -= c[i][curr+1];  
 assign[curr+1] = 0;  
 }  
 }  
}

