算法作业第 6 章

毛翰翔　　210110531

2022 年 12 月 4 日

1、用本章知识解决下面的问题，写出你的思路和伪代码。

在商店中，有许多在售的物品。然而，也有一些大礼包，每个大礼包以优惠的价格捆绑销售一组物品。现给定每个物品的价格，每个大礼包包含物品的清单，以及待购物品清单。请输出确切完成待购清单的最低花费。每个大礼包由一个数组中的一组数据描述，最后一个数字代表大礼包的价格，其他数字分别表示内含的其他种类物品的数量。任意大礼包可无限次购买。

示例 1:

输入: [2,5], [[3,0,5],[1,2,10]], [3,2]

输出: 14

解释:

有 A 和 B 两种物品，价格分别为 ¥2 和 ¥5。

大礼包 1，你可以以 ¥5 的价格购买 3A 和 0B。

大礼包 2，你可以以 ¥10 的价格购买 1A 和 2B。

你需要购买 3 个 A 和 2 个 B，所以你付了 ¥10 购买了 1A 和 2B（大礼包 2），以及 ¥4 购买 2A。

示例 2:

输入: [2,3,4], [[1,1,0,4],[2,2,1,9]], [1,2,1]

输出: 11

解释:

A，B，C 的价格分别为 ¥2，¥3，¥4.

你可以用 ¥4 购买 1A 和 1B，也可以用 ¥9 购买 2A，2B 和 1C。

你需要买 1A，2B 和 1C，所以你付了 ¥4 买了 1A 和 1B（大礼包 1），以及 ¥3 购买 1B，¥4 购买 1C。

你不可以购买超出待购清单的物品，尽管购买大礼包 2 更加便宜

说明:

最多 6 种物品，100 种大礼包。

每种物品，你最多只需要购买 6 个。

你不可以购买超出待购清单的物品，即使更便宜。

Solution：基本思路是使用深度优先遍历. 使用递归函数.

我们首先考察没有使用礼包的情况, 将其和使用了礼包的进行对比. 前者是能够求出的, 后者调用函数求出来.

设这个函数是 `dfs` , 假设使用了第 i 礼包, 那么对 needs 数组进行更新, 对于使用了礼包之后剩余的需求, 我们使用类似的方法, 先是考察不使用礼包的情况, 再考察使用了第 j 个礼包的情况.

其中, i , j 都是任意的, 于是说每次递归调用都需要 NumofSpecial 次, 其中 NumofSpecial 是礼包的个数.

```
#define infty 100000
int dfs (int CurrentSpecial , int *ShinNeeds, int **special, int n, int *price, int NumofSpecial)
    int CopyNeeds[10] = {0};    // cpoy 一个数组，向下进行一个传递.
    int temp = 0, ans = 0;
    for (int i  = 0 ; i < n ; i++){
        CopyNeeds[i] = ShinNeeds[i] - special[CurrentSpecial][i];
                                // 根据这个礼包，更新这个 needs 数组
        if (CopyNeeds[i] < 0)   // 如果说不用买这个礼包就直接返回最大值.
        return infty;
    }
    // ans 更新为不买礼包的情况.
    for (int i = 0 ; i < n ; i++)
        ans = ans + price[i] * CopyNeeds[i];
    // 遍历买各种礼包的情况.
    for (int i = 0 ; i < NumofSpecial ; i++){
        temp = dfs(i, CopyNeeds, special, n, price, NumofSpecial);
        if (temp < ans)
    // 如果说 dfs 结果更低一些，就将 ans 更新.
            ans = temp;
    }
    return ans + special[CurrentSpecial][n];
}
int shoppingOffers(int* price, int priceSize, int** special, int specialSize, int* specialColSize
    int ans = infty;                // ans for answer
    int NumofSpecial = specialSize; // NumofSpecial , the number of specials
    int n = priceSize;              // n for number of items
    int temp;
    for (int i = 0; i < NumofSpecial; i++){
        for (int j = 0; j < n+1; j++)
            scanf("%d", &special[i][j]);
    }
    // ans 更新为直接购入
    for (int i = 0; i < n; i++)
        ans = ans + needs[i] * price[i];
    //
    for (int i = 0 ; i < NumofSpecial ; i++){
        temp = dfs(i , needs,special , n , price , NumofSpecial);
        if (temp < ans)
        ans = temp;
    }
    return ans;
}
```

2. 给定一个 4 个点的连通有向图，其邻接矩阵如下:

$$\begin{bmatrix} \infty & 9 & 13 & 15 \\ 2 & \infty & 1 & 4 \\ 3 & 5 & \infty & 1 \\ 9 & 6 & 3 & \infty \end{bmatrix}$$

可用使用 A* 算法求这个图的旅行商问题。

(1) 请写出你的 g(n) 和 h*(n) 的定义。

(2) 画出求解此图的搜索树。

solution: $g(n)$ 的定义是, 给定一个起点, 这个起点到 n 的距离, $h^*(n)$ 的定义是, n 经过剩下的顶点回到起点的最优值.

类似的, 这里使用 “连接了 n 和未经过的节点的边” 中最低权重. 作为 $h^*(n)$ 的一个下界.

这里我们使用 1 为起点.

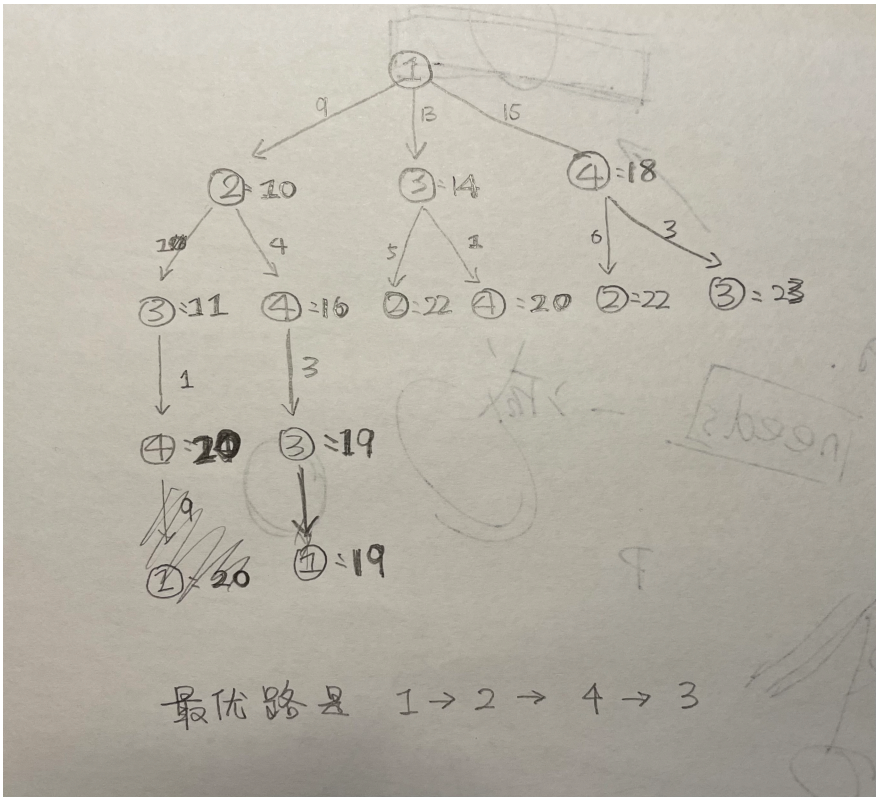

图 1: 树