数据结构与算法I 作业15

2019201409 于倬浩

16.1-5

本题与原始问题的唯一差异在于，每个被选中的区间对答案的贡献从1变成了各自的权值。对于原问题，贪心算法的正确性证明时，依赖区间权值为1的性质，在划分子问题时做到了决策的包容性，即如果在一个子问题的最优解中选择了当前结束最早的区间，那么这个区间一定会在子问题的扩展问题中也被选择，理由是不会使答案变得更差。然而，如果不同区间的权值不同，那么这样的包容性显然不再存在，因为如果子问题中的最优解与一个扩展问题中权值更大的区间发生冲突，那么子问题中的最优解就会被替换掉，即贪心策略不可行。

因此，本题依然需要使用动态规划算法。设`MaxIntervals(L,R)`表示当前只选择被`[L,R]`这段区间包含的区间，所能实现的最优解。那么假设当前问题的最优解中包含了区间`[a,b]`，则子问题为`MaxIntervals(L,a)`, `MaxIntervals(b,R)`，那么由定义可知，将三者取并后即可实现当前的最优解，即这种定义方式可以保证最优子结构。

具体实现上，可以将所有区间按照结束时间从早到晚排序，设`f[i]`表示当前选中的所有区间中最晚的结束时间为`i`的最优解。那么从小到大枚举`i`，枚举最后一个被选择的区间`j`，如果`j`的结束时间比`i`开始时间早，那么有转移：$f[i] = max_j(f[j]) + w_i$。

由于状态数为$\Theta(n)$，单次转移代价，在不使用任何优化的情况下为$\Theta(n)$，因此时间复杂度为$\Theta(n^2)$，空间复杂度$\Theta(n)$。

注意到对于每个`i`，实际上可以选择的`j`可以通过二分计算出最大下标，那么只需多维护一个$h[i] = max_{j=1}^{i}(f[i])$，接着在转移时只要二分到最大的`j`，即可。单次转移代价为$\Theta(lgn)$，因此时间复杂度为$\Theta(nlgn)$，空间复杂度$\Theta(n)$。

代码如下：

```
struct Inverval{
    int l, r, v; //区间的左右端点、权值
    inline bool operator < (const Inverval &b) {
        return r < b.r; //按照右端点排序
    }
}s[maxn];
int n, f[maxn], h[maxn];
int main() {
    cin >> n;
    for(int i = 1; i ≤ n; ++i) {
        int l, r, v;
        cin >> l >> r >> v;
        s[i] = (Inverval){l, r, v};
    }
    sort(&s[1], &s[n + 1]);
    for(int i = 1; i ≤ n; ++i) {
        if(s[i].l < s[1].r) { //当前区间只能作为第一个区间出现
            f[i] = s[i].v;
            h[i] = max(f[i], h[i - 1]);
        }
        else { //当前区间之前可以有其他区间
            int L = 1, R = i - 1, M;
            while(R > L) { //二分找出最大的j
                M = L + R + 1 >> 1;
                if(s[M].r ≤ s[i].l) L = M;
                else R = M - 1;
            }
            f[i] = h[L] + s[i].v;
            h[i] = max(h[i - 1], f[i]);
        }
    }
    cout << h[n] << endl;
}
```

测试结果：

```
5
1 7 10
1 2 1
3 5 11
4 5 100
4 7 9

101
```

16.2-4

将问题抽象如下：数轴的正半轴上有n个从小到大有序的点，且已知这些点的坐标x_i。教授从原点出发，每次可以行驶的距离不超过m，问到达坐标最大的点需要停车几次。

贪心策略如下：如果当前行驶至x_i，且目前距上次停车位置的行驶距离$+x_{i+1}-x_i$大于m，则在当前点停车，否则继续行驶。即每次停车后，都尽量不停车，直到无法到达下一个点时，才在当前点停车。

考虑按照贪心策略，第一次停车的位置是x_p，那么实际上对应的子问题即为从x_p出发，到x_n的最优方案。即按照这种贪心策略，每个问题只有一个对应的子问题。

接下来考虑最优性质：假设按照贪心策略，第一次停车的位置是x_p。任取一个最优解$Q=x_{i1},x_{i2}\ldots,x_{ik}$。那么首先由贪心策略可知，$x_{i1}\leq x_p$，因为$x_p$是我们从起点出发能到达的最远点。接下来，如果使用$x_p$代替$x_{i1}$，答案显然不会变得更差，因为$x_{i2}-x_p\leq x_{i2}-x_{i1}$，即$x_p$一定可以替换任意一组最优解的第一个位置。接下来递归考虑，即可证明，任意一组最优解都可以按照这样的方法，按照贪心策略调整，同时保证解的最优性，因此贪心算法的最优性得证。

代码如下：

```
int ans = 0;
for(int i = 1, j; i < n;) {
    j = i + 1, ++ans;
    while(j < n && x[j + 1] - x[i] <= m) ++j;
    i = j;
}
```

时间复杂度$\Theta(n)$，空间复杂度$\Theta(n)$。