动态规划—DP的转化 学习笔记（待更）

动态规划与最短路

例题：P2761 软件补丁问题，很容易写出转移方程：$dp_s \leftarrow dp_{s \setminus F_1 \cup F_2} + t_i$，

但是这样就出现了环，没有形成 DAG 就无法跑动态规划了，怎么办？

可以将原问题转换为［最短路］：

将原状态 s 记为一个点，将原转移路径记为一条边 (s, s')，然后跑最短路即可。

这种问题的转移方程，形如：$f_v = f_u + w$，即有边 $(u \to v, w)$。

当然前提是状态数不能太多，因为最短路的复杂度为 $O(n^2)$，或 $O(m \log n)$，根据情况选择。

代码：

```
typedef pair<int, int> PII;

const int N = 21, M = 110;

int n, m, t[M];
int b1[M], b2[M], f1[M], f2[M];

int cuse(int s, int i) {
    if ((s & b1[i]) == b1[i] && (s & b2[i]) == 0) return (s & ~f1[i]) |
f2[i];
    return -1;
}

int dis[1 << N];
bool st[1 << N];

int dijkstra(int s, int e) {
    memset(dis, 0x3f, sizeof dis); const int INF = dis[0];
    priority_queue<PII, vector<PII>, greater<PII>> heap;
```

```
    dis[s] = 0; heap.push({0, s});
    while (heap.size()) {
        int u = heap.top().second, d = heap.top().first, v; heap.pop();
        if (st[u]) continue; st[u] = true;
        for (int i = 0; i < m; ++i) {
            if ((v = cuse(u, i)) == -1) continue;
            if (dis[v] > d + t[i]) {
                dis[v] = d + t[i];
                heap.push({dis[v], v});
            }
        }
    }
    return dis[e] == INF ? 0 : dis[e];
}

signed main() {
    n = ur, m = ur; char b[N], f[N];
    for (int i = 0; i < m; ++i) {
        t[i] = ur; scanf("%s %s", b, f);
        for (int j = 0; j < n; ++j) {
            if (b[j] == '+') b1[i] |= 1 << j;
            else if (b[j] == '-') b2[i] |= 1 << j;
            if (f[j] == '-') f1[i] |= 1 << j;
            else if (f[j] == '+') f2[i] |= 1 << j;
        }
    }
    printf("%lld\n", dijkstra((1 << n) - 1, 0));
    return 0;
}
```

合集：学习笔记

标签：算法 , 学习笔记