背包与树形动态规划

2019年1月25日

黄哲威 hzwer
北京大学16级计算机科学

第二节目标

- 背包问题
- 注：本课件背包问题部分参考崔添翼《背包问题九讲 2.0 RC1》，https://github.com/tianyicui/pack
- 树的基本概念
- 树的存储遍历
- 树的信息统计，树上的递推，处理技巧

01背包问题

- 有 N 件物品和一个容量为 V 的背包。放入第 i 件物品耗费的空间是 Ci，得到的价值是 Wi。求解将哪些物品装入背包可使价值总和最大。
- 这是一个NPC问题。

01背包问题

– F[i, v] 表示前 i 件物品恰放入一个容量为 v 的背包可以获得的最大价值。则其状态转移方程便是：

– $F[i, v] = \max\{F[i-1, v], F[i-1, v-C_i] + W_i\}$

01背包问题

– F[i, v] 表示前 i 件物品恰放入一个容量为 v 的背包可以获得的最大价值。则其状态转移方程便是：
– $F[i, v] = \max\{F[i-1, v], F[i-1, v-C_i] + W_i\}$
– “将前 i 件物品放入容量为 v 的背包中”这个子问题，若只考虑第 i 件物品的策略（放或不放），那么就可以转化为一个只和前 i − 1 件物品相关的问题。
– 如果不放第 i 件物品，那么问题就转化为“前 i − 1 件物品放入容量为 v 的背包中”，价值为 $F[i-1, v]$；如果放第 i 件物品，那么问题就转化为“前 i − 1 件物品放入剩下的容量为 $v - C_i$ 的背包中”，此时能获得的最大价值就是 $F[i-1, v-C_i]$ 再加上通过放入第 i 件物品获得的价值 W_i。

01背包问题

– F[i, v] 表示前 i 件物品恰放入一个容量为 v 的背包可以获得 的最大价值。则其状态转移方程便是：

$$
\begin{aligned}
& F[0, 0..V] \leftarrow 0 \\
& \texttt{for } i \leftarrow 1 \texttt{ to } N \\
& \qquad \texttt{for } v \leftarrow 0 \texttt{ to } C_i - 1 \\
& \qquad\qquad F[i, v] \leftarrow F[i-1, v] \\
& \qquad \texttt{for } v \leftarrow C_i \texttt{ to } V \\
& \qquad\qquad F[i, v] \leftarrow \max\{F[i-1, v], F[i-1, v - C_i] + W_i\}
\end{aligned}
$$

01背包问题

- $F[i, v] = \max\{F[i-1, v], F[i-1, v-C_i] + W_i\}$
- 我们希望能只用一维数组解决这个问题

01背包问题

- $F[i, v] = \max\{F[i-1, v], F[i-1, v-C_i] + W_i\}$
- 我们希望能只用一维数组解决这个问题
- $F[i, v]$ 是由 $F[i-1, v]$ 和 $F[i-1, v-C_i]$ 两个子问题递推而来，能否保证在推 $F[i, v]$ 时（也即在第 i 次主循环中推 $F[v]$ 时）能够取用 $F[i-1, v]$ 和 $F[i-1, v-C_i]$ 的值呢?

01背包问题

– F[i, v] = max{F[i − 1, v], F[i − 1, v − Ci] + Wi}

$$
\begin{aligned}
& F[0..V] \leftarrow 0 \\
& \texttt{for } i \leftarrow 1 \texttt{ to } N \\
& \qquad \texttt{for } v \leftarrow V \texttt{ to } C_i \\
& \qquad\qquad F[v] \leftarrow \max\{F[v], F[v - C_i] + W_i\}
\end{aligned}
$$

完全背包问题

– 有 N 种物品和一个容量为 V 的背包，每种物品都有无限件可用。放入第 i 种物品 的费用是 Ci，价值是 Wi。求解：将哪些物品装入背包，可使这些物品的耗费的费用总 和不超过背包容量，且价值总和最大。

完全背包问题

- 有 N 种物品和一个容量为 V 的背包，每种物品都有无限件可用。放入第 i 种物品 的费用是 Ci，价值是 Wi。求解：将哪些物品装入背包，可使这些物品的耗费的费用总 和不超过背包容量，且价值总和最大。
- 这个问题非常类似于 01 背包问题，所不同的是每种物品有无限件。也就是从每种物品的角度考虑，与它相关的策略已并非取或不取两种，而是有取 0 件、取 1 件、取 2 件……直至取 $\lfloor V/C_i \rfloor$ 件等许多种。

完全背包问题

- 仍然按照解 01 背包时的思路，令 F[i, v] 表示前 i 种物品恰放入一个容量为 v 的背包的最大权值。
- $F[i, v] = \max\{F[i-1, v-kC_i] + kW_i \mid 0 \le kC_i \le v\}$

完全背包问题

- 仍然按照解 01 背包时的思路，令 $F[i, v]$ 表示前 i 种物品恰放入一个容量为 v 的背包的最大权值。
- $F[i, v] = \max\{F[i-1, v-kC_i] + kW_i \mid 0 \le kC_i \le v\}$
- 这跟 01 背包问题一样有 $O(VN)$ 个状态需要求解，但求解每个状态的时间不是常数，求解状态 $F[i, v]$ 的时间是 $O(v/C_i)$，总的复杂度可以认为是 $O(NV \Sigma (V/C_i))$。

完全背包问题

– 我们可以考虑把完全背包问题转化为 01 背包问题来解。
– 最简单的想法是，考虑到第 i 种物品最多选 $\lfloor V/C_i \rfloor$ 件，于是可以把第 i 种物品转 化为 $\lfloor V/C_i \rfloor$ 件费用及价值均不变的物品，然后求解这个 01 背包问题。这样的做法完 全没有改进时间复杂度，但这种方法也指明了将完全背包问题转化为 01 背包问题的思路：将一种物品拆成多件只能选 0 件或 1 件的 01 背包中的物品。

完全背包问题

– 我们可以考虑把完全背包问题转化为 01 背包问题来解。

– 更高效的转化方法是：把第 i 种物品拆成费用为 $Ci2^k$、价值为 $Wi2^k$ 的若干件物品，其中 k 取遍满足 $Ci2^k \le V$ 的非负整数。因为不管最优策略选几件第 i 种物品，其件数写成二进制后，总可以表示成若干个 2^k 件物品的和。这样一来就把每种物品拆成 $O(\log \lfloor V/Ci \rfloor)$ 件物品。

完全背包问题

– 首先想想为什么 01 背包中要按照 v 递减的次序来循环让 v 递减是为了保证第 i 次循环中的状态 F[i, v] 是由状态 F[i − 1, v − Ci] 递推而来。换句话说，这正是为了保证每件物品只选一次，保证在考虑“选入第 i 件物品”这件策略时，依据的是一个绝无已经选入第 i 件物品的子结果 F[i − 1, v − Ci]。
– 而现在完全背包的特点恰是每种物品可选无限件，所以在考虑“加选一件第 i 种物品”这种策略时， 却正需要一个可能已选入第 i 种物品的子结果 F[i, v − Ci]，所以就可以并且必须采用 v 递增的顺序循环。

完全背包问题

$$
\begin{array}{l}
F[0..V] \leftarrow 0 \\
\texttt{for } i \leftarrow 1 \texttt{ to } N \\
\quad \texttt{for } \underline{v \leftarrow V \texttt{ to } C_i} \\
\quad\quad F[v] \leftarrow \max\{F[v], F[v-C_i]+W_i\}
\end{array}
$$

$$
\begin{array}{l}
F[0..V] \leftarrow 0 \\
\texttt{for } i \leftarrow 1 \texttt{ to } N \\
\quad \texttt{for } \underline{v \leftarrow C_i \texttt{ to } V} \\
\quad\quad F[v] \leftarrow \max(F[v], F[v-C_i]+W_i)
\end{array}
$$

多重背包问题

- 有 N 种物品和一个容量为 V 的背包。第 i 种物品最多有 Mi 件可用，每件耗费的空间是 Ci，价值是 Wi。求解将哪些物品装入背包可使这些物品的耗费的空间总和不超过背包容量，且价值总和最大。

多重背包问题

- 有 N 种物品和一个容量为 V 的背包。第 i 种物品最多有 Mi 件可用，每件耗费的空间是 Ci，价值是 Wi。求解将哪些物品装入背包可使这些物品的耗费的空间总和不超过背包容量，且价值总和最大。
- 基本的方程只需将完全背包问题的方程略微一改即可。因为对于第 i 种物品有 Mi + 1 种策略：取 0 件，取 1 件……取 Mi 件。令 F[i, v] 表示前 i 种物品恰放入一个容量为 v 的背包的最大价值，则有状态转移方程：
- $F[i, v] = \max\{F[i-1, v-k*C_i] + k*W_i \mid 0 \le k \le M_i\}$
- 复杂度是 $O(V \Sigma M_i)$。

多重背包问题

– 多重背包也可以转化成 01 背包问题：把第 i 种物品换成 Mi 件 01 背包中的物品，则得到了物品数为 ΣMi 的 01 背包问题。
– 仍然考虑二进制的思想，我们考虑把第 i 种物品换成若干件物品，使得原问题中第 i 种物品可取的每种策略——取 0 . . . Mi 件——均能等价于取若干件代换以后的物品。 另外，取超过 Mi 件的策略必不能出现。
– 将第 i 种物品分成若干件 01 背包中的物品，其中每件物品有一个系数。这件物品的费用和价值均是原来的费用和价值乘以这个系数。令这些系数分别为 1, 2, 2^2 . . . 2^(k−1) , Mi − 2^k + 1，且 k 是满足 Mi − 2^k + 1 > 0 的最大整数。例如，如果 Mi 为 13，则相应的 k = 3，这种最多取 13 件的物品应被分成系数分别为 1, 2, 4, 6 的四件物品。
– 这样就将第 i 种物品分成了 O(logMi) 种物品，将原问题转化为了复杂度为O(V ΣlogMi) 的 01 背包问题，是很大的改进。

多重背包问题

- 多重背包也可以转化成 01 背包问题：把第 i 种物品换成 Mi 件 01 背包中的物品，则得到了物品数为 ΣMi 的 01 背包问题。

二维费用的背包问题

- 二维费用的背包问题是指：对于每件物品，具有两种不同的费用，选择这件物品必须同时付出这两种费用。对于每种费用都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。 设第 i 件物品所需的两种费用分别为 C_i 和 D_i。两种费用可付出的最大值（也即两种背包容量）分别为 V 和 U。物品的价值为 W_i。

二维费用的背包问题

- 二维费用的背包问题是指：对于每件物品，具有两种不同的费用，选择这件物品必须同时付出这两种费用。对于每种费用都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。 设第 i 件物品所需的两种费用分别为 C_i 和 D_i。两种费用可付出的最大值（也即两种背包容量）分别为 V 和 U。物品的价值为 W_i。
- 费用加了一维，只需状态也加一维即可。设 $F[i, v, u]$ 表示前 i 件物品付出两种费用 分别为 v 和 u 时可获得的最大价值。状态转移方程就是：
- $F[i, v, u] = \max\{F[i-1, v, u], F[i-1, v-C_i, u-D_i] + W_i\}$

二维费用的背包问题

- 二维费用的背包问题是指：对于每件物品，具有两种不同的费用，选择这件物品必须同时付出这两种费用。对于每种费用都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。 设第 i 件物品所需的两种费用分别为 Ci 和 Di。两种费用可付出的最大值（也即两种背包容量）分别为 V 和 U。物品的价值为 Wi。
- 费用加了一维，只需状态也加一维即可。设 F[i, v, u] 表示前 i 件物品付出两种费用 分别为 v 和 u 时可获得的最大价值。状态转移方程就是：
- $F[i, v, u] = \max\{F[i-1, v, u], F[i-1, v-C_i, u-D_i] + W_i\}$
- 如前述优化空间复杂度的方法，可以只使用二维的数组：当每件物品只可以取一次 时变量 v 和 u 采用逆序的循环，当物品有如完全背包问题时采用顺序的循环，当物品 有如多重背包问题时拆分物品。

分组的背包问题

– 有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 Ci，价值是 Wi。这些物品被划分为 K 组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

分组的背包问题

- 有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 Ci，价值是 Wi。这些物品被划分为 K 组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
- 这个问题变成了每组物品有若干种策略：是选择本组的某一件，还是一件都不选。 也就是说设 F[k, v] 表示前 k 组物品花费费用 v 能取得的最大权值，则有：
- $F[k, v] = \max\{F[k-1, v], F[k-1, v-C_i] + W_i \mid \text{item } i \in \text{group } k\}$
- 时间复杂度O(NV)

有依赖的背包问题

– 这种背包问题的物品间存在某种“依赖”的关系。也就是说，物品 i 依赖于物品 j， 表示若选物品 i，则必须选物品 j。为了简化起见，我们先设没有某个物品既依赖于别的物品，又被别的物品所依赖；另外，没有某件物品同时依赖多件物品。

有依赖的背包问题

- 这种背包问题的物品间存在某种“依赖”的关系。也就是说，物品 i 依赖于物品 j， 表示若选物品 i，则必须选物品 j。为了简化起见，我们先设没有某个物品既依赖于别的物品，又被别的物品所依赖；另外，没有某件物品同时依赖多件物品。
- 将不依赖于别的物品的物品称为“主件”，依赖于某主件的物品称为“附件”。可知所有的物品由若干主件和依赖于每个主件的一个附件集合组成。

有依赖的背包问题

- 这种背包问题的物品间存在某种“依赖”的关系。也就是说，物品 i 依赖于物品 j， 表示若选物品 i，则必须选物品 j。为了简化起见，我们先设没有某个物品既依赖于别的物品，又被别的物品所依赖；另外，没有某件物品同时依赖多件物品。
- 将不依赖于别的物品的物品称为“主件”，依赖于某主件的物品称为“附件”。可知所有的物品由若干主件和依赖于每个主件的一个附件集合组成。
- 可以对主件 k 的 “附件集合”先进行一次 01 背包，得到费用依次为 $0...V - C_k$ 所有这些值时相应的最大价值 $F_k[0...V - C_k]$。那么，这个主件及它的附件集合相当于 $V - C_k + 1$ 个物品的 物品组，其中费用为 v 的物品的价值为 $F_k[v - C_k] + W_k$，v 的取值范围是 $C_k \le v \le V$ 。
- 这样就转化为一个分组背包的问题。

泛化物品

- 考虑这样一种物品，它并没有固定的费用和价值，而是它的价值随着你分配给它的 费用而变化。这就是泛化物品的概念。
- 在背包容量为 V 的背包问题中，泛化物品是一个定义域为 0 . . . V 中的整数的函数 h，当分配给它的费用为 v 时，能得到的价值就是 h(v)。

泛化物品

– 一个费用为 c 价值为 w 的物品，如果它是 01 背包中的物品，那么把它看成泛化物 品，它就是除了 $h(c) = w$ 外，其它函数值都为 0 的一个函数。如果它是完全背包中的物品，那么它可以看成这样一个函数，仅当 v 被 c 整除时有 $h(v) = w \cdot v / c$，其它函数值均为 0。如果它是多重背包中重复次数最多为 n 的物品，那么它对应的泛化物品的函 数有 $h(v) = w \cdot v / c$ 仅当 v 被 c 整除且 $v / c \le n$，其它情况函数值均为 0。

– 一个物品组可以看作一个泛化物品 h。对于一个 $0 . . . V$ 中的 v，若物品组中不存在 费用为 v 的物品，则 $h(v) = 0$，否则 $h(v)$ 取值为所有费用为 v 的物品的最大价值。

泛化物品的和

– 如果给定了两个泛化物品 h 和 l，要用一定的费用从这两个泛化物品中得到最大的价值，这个问题怎么求呢?

泛化物品的和

- 如果给定了两个泛化物品 h 和 l，要用一定的费用从这两个泛化物品中得到最大的价值，这个问题怎么求呢?
- 事实上，对于一个给定的费用 v，只需枚举将这个费用如何分配给两个泛化物品就可以了。同样的，对于 0. . .V 中的每一个整数 v，可以求得费用 v 分配到 h 和 l 中的最大价值 f(v)。也即
- $f(v) = \max\{h(k) + l(v - k) \mid 0 \le k \le v\}$泛化物品和运算的时间复杂度取决于背包的容量，是O(V^2)。
- 由泛化物品的定义可知：在一个背包问题中，若将两个泛化物品代以它们的和，不影响问题的答案。

练习题

BZOJ2287. 消失之物

» ftiasch 有 *N* 个物品, 体积分别是 *W1*, *W2*, …, *WN*。 由于她的疏忽, 第 *i* 个物品丢失了。“要使用剩下的 *N* – 1 物品装满容积为 *x* 的背包，有几种方法呢？” — 这是经典的问题了。她把答案记为 *Count(i, x)* ，想要得到所有1 <= i <= N, 1 <= x <= M 的 *Count(i, x)* 表格。

» $(1 \le N \le 2 \times 10^3)$ $(1 \le M \le 2 \times 10^3)$

BZOJ2287. 消失之物

- f(i,j)表示使用前i个物品填满j的空间的方案数：
- F(i,j)=f(i-1,j)+f(i-1,j-w(i)),f(0,0)=1

BZOJ2287. 消失之物

- f(i,j)表示使用前i个物品填满j的空间的方案数：
- F(i,j)=f(i-1,j)+f(i-1,j-w(i)),f(0,0)=1
- c(i,j)表示count(i,j),分三种情况
- j>=w(i),c(i,j)=f(n,j)-c(i,j-w(i))即用填满j的总方案数（f(n,j)）减去使用了第i个物品的方案数，其中使用第i个物品填满j的空间的方案数等于使用其余物品填满j-w(i)的空间的方案数(c(i,j-w(i)))
- 0<j<w(i),c(i,j)=f(n,j)，此时无论怎么填，都不会用到第i个物品，答案所以等于总方案数。
- j=0,c(i,j)=1

认识树

- 树是一种十分优美的数据结构，因为它本身就具有的递归性，所以树和子树之间能相互传递很多信息。
- 把它叫做“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上，而叶朝下的。
- 树上的许多特征都可以通过它的子树的对应特征计算获得。
- 所以树做动态规划求最优解和做统计非常方便。

树的概念

- n个点，n-1条边的无向连通图称为树
- 节点的度：一个节点含有的子树的个数称为该节点的度
- 叶节点或终端节点：度为0的节点称为叶节点
- 父节点：若一个节点含有子节点，则这个节点称为其子节点的父节点
- 子节点：一个节点含有的子树的根节点称为该节点的子节点
- 兄弟节点：具有相同父节点的节点互称为兄弟节点

树的概念

» 树的度：一棵树中，最大的节点的度称为树的度

» 节点的层次：从根开始定义起，根为第1层，根的子节点为第2层，以此类推

» 树的高度或深度：树中节点的最大层次

» 节点的祖先：从根到该节点所经分支上的所有节点

» 子孙：以某节点为根的子树中任一节点都称为该节点的子孙

» 森林：由m（m>=0）棵互不相交的树的集合称为森林

例题 1

- 给一棵 n 个点的无权树，问树中每个子树的大小，每个节点的深度?
- 其中 $0 \le n \le 10^5$

例题 1

» 给一棵 n 个点的无权树，问树中每个子树的大小，每个节点的深度?

» 其中 $0 \leq n \leq 10^5$

```
vector<int>v[100005];
void getsize(int x)
{
   size[x]=1;
   for (int i=0;i<v[x].size();i++)
   {
      getsize(v[x][i]);
      size[x]+=size[v[x][i]];
      mx_size[x]=max(mx_size[x],size[v[x][i]]);
   }
}
```

例题 1

- 给一棵 n 个点的无权树，问树中每个子树的大小，每个节点的深度?
- 其中 $0 \le n \le 10^5$

```
vector<int>v[100005];
```

```
void getdep(int x)
{
    for (int i=0;i<v[x].size();i++)
    {
        dep[v[x][i]]=dep[x]+1;
        getdep(v[x][i]);
    }
}
```

例题 2

» 给一棵 n 个点的点权树，问树中每个子树的点权和，点权最大值?

» 其中 $0 \le n \le 10^5$

```
void getmx(int x)
{
   mx[x]=val[x];
   for(int i=0;i<v[x].size();i++)
   {
      getmx(v[x][i]);
      mx[x]=max(mx[x],mx[v[x][i]]);
   }
}
```

例题 3

- 给一棵 n 个点的无权树，求树的重心? 重心定义为，删去该点之后，图中的所有连通块的最大尺寸最小。
- 其中 $0 \le n \le 10^5$

例题 3

- 给一棵 n 个点的无权树，求树的重心? 重心定义为，删去该点之后，图中的所有连通块的最大尺寸最小。
- 其中 $0 \le n \le 10^5$
- 用size[x]表示x子树的结点数
- $size[x]=\sum size[yi]$
- mx[x]表示删去x之后的最大子树结点数
- $mx[x]=\max\{size[yi],n-size[x]\}$

例题 4

- 给一棵 n 个点的边权树，问树中每个子树的最长链?次长链?
- 其中 $0 \le n \le 10^5$

例题 4

- 给一棵 n 个点的边权树，问树中每个子树的最长链?次长链? （只考虑从根）
- 其中 $0 \le n \le 10^5$
- f1[x], f2[x]表示以 x 为根的树的最长链
- f2[x] = f1[x], f1[x] = f1[yi] + e (f1[yi] + e > f1[x])
- f2[x] = max(f2[x], f2[yi] + e) else

例题 5

» 给一棵 n 个点的边权树，求树的直径。

» 树的直径一定为某个点到其不同子树叶子的最长链 + 次长链

» 其中 $0 \le n \le 10^5$

例题 5

- 树的直径一定为某个点到其不同子树叶子的最长链 + 次长链
- 利用之前的dp结果

例题 5

- 还有一个做法。
- 选择一个点 X，求出其最远点 Y，再求 Y 的最远点 Z，则 YZ 为 直径
- 这样为什么是对的?

例题 5

» 分三步证明，反证法。

» 先证明 XY 与直径有交

» 再证明 Y 是直径的一个端点

» 那么找 Y 的最远点 Z，YZ 即为直径

例题 6

- 给一棵 n 个点的无权树，求其每个子树的重心。
- 子树重心定义为，删去该点之后，子树的所有连通块大小均不超过 $n/2$
- 其中 $0 \le n \le 10^5$

例题 6

» 用yi表示x结点的儿子

» 预处理size[x]和mx[x]表示树的大小，yi树的最大结点数，用于判断一个结点是不是重心

» 根据性质，x的重心应该在yi的重心之间的路径上

» 转移的时候只要把每个树yi的重心到x的路径从下往上暴力判断一下，发现重心就退出

» 复杂度相当于从每个叶子走向根，也就是遍历一棵树，显然是O（n）的

例题 7.没有上司的舞会

» 公司的人际关系构成一棵 n 个结点的树，现公司要举行一场晚会

» 并规定如果邀请了某个人，那么一定不会邀请他的上司(上司的上司， 上司的上司的上司......都可以邀请)。

» 每个人都有一个气氛值，求一个邀请方案，使欢乐值的和最大。

» 其中 $0 \le n \le 10^5$

例题 7.没有上司的舞会

- f_i表示 i 人参加了舞会的时候这个子树的欢乐值之和
- g_i表示 i 人没参加舞会的时候这个子树的欢乐值之和

$$f_i = v_i + \sum g_j (fa_j = i)$$
$$g_i = \sum max(f_j, g_j)(fa_j = i)$$

例题 8. 二叉苹果树

- 有一棵苹果树，它是一棵二叉树，共 N 个节点，树节点编号为 1~N
- 编号为 1 的节点为树的根，边可理解为树的分枝，每个分支都长着若干个苹果
- 现在要减去若干个分支，保留 M 个分支，使得这 M 个分支中的 苹果数量最多。
- 其中 $0 \le n \le 10^4$, $0 \le m \le 50$

例题 8. 二叉苹果树

» 此题和前面的题有个明显不同的地方，我们关注树的信息，还需要在树上分配资源。

» 所以我们描述问题时需增加一维表示要分配的资源, 用 f[i, j] 表示 以 i 为根的树上保留 j 个边能获得的最多的苹果个数。

» 我们可以用子树的相关特性算出树的相关特性。

$$f[i,j] = max(f[sonl, j-1] + W[i, sonl],$$
$$f[sonr, j-1] + W[i, sonr],$$
$$f[sonl, k] + f[sonr, j-2-k] + W[i, sonl] + W[i, sonr])$$
$$(0 \le k \le j-2)$$

W 表示对应树枝上的苹果数。

答案是 $f[root, M]$

时间复杂度是 $O(NM^2)$

例题 8. 二叉苹果树

- 此题和前面的题有个明显不同的地方，我们关注树的信息，还需要在树上分配资源。
- 所以我们描述问题时需增加一维表示要分配的资源, 用 $f[i, j]$ 表示 以 i 为根的树上保留 j 个边能获得的最多的苹果个数。
- 我们可以用子树的相关特性算出树的相关特性。

例题 9. Sta

- 给一个 n 个点的边权树，求其一个根，使得所有点的深度和最大。
- 其中 $0 \leq n \leq 10^5$

例题 9. Sta

发现从根从某个位置移到它的一个子树得出 ans 只要 O(1) 的时间

例题 10. 奶牛大集会

- 给一棵 n 个点的边 + 点权树，求带权重心。

- 其中 $0 \le n \le 10^5$

例题 10. 奶牛大集会

» 先递推预处理出size[x]与dis[x]，表示子树x的大小，树根到x的距离，tot表示整颗树的大小

» 一开始假设集会的点在根，先得到一个ans

» 若把集会的地点从x移动到它的儿子y，设经过的边为i，权为wi，则

$$ans' = ans - size_y * w_i + (tot - size_y) * w_i$$

$$ans' = ans + (tot - 2 * size_y) * w_i$$

» 显然每次符合条件的儿子只可能有一个，可以预处理后贪心得到最优解

例题 11. 电话网络

- 一棵 n 个点的无权树，求最小点覆盖。
- 其中 $0 \le n \le 10^5$

例题 11. 电话网络

» 树形 dp 考虑一个点被儿子/自己/父亲覆盖
f(i,0) 表示以 i 为根的子树中所有点均被覆盖且草地 i 上无信号塔所需的最小塔数(i 被其儿子覆盖)

» f(i,1) 以 i 为根的子树中所有点均被覆盖且草地 i 上有信号塔所需的最小塔数

» f(i,2) 以 i 为根的子树中除 i 点以外其余点均被覆盖所需的最小塔数

» 其实可以贪心

Q & A