搜索 & 剪枝

quarter

2018 年 11 月 8 日

首先

~~迷之又双叒叕要讲这个专题。~~

没东西讲了.jpg

Description

蛋糕形状为一层层的圆柱体，自底向上直径递减。

要求制作一个体积为 $N\pi$ 的 M 层蛋糕。设从下往上数第 $i(1 \le i \le M)$ 层蛋糕是半径为 R_i, 高度为 H_i 的圆柱。当 $i < M$ 时，要求 $R_i > R_{i+1}$ 且 $H_i > H_{i+1}$。令外表面面积为 $S\pi$。

对于给出的 N 和 M，找出一个制作方案（适当的 R_i 和 H_i 的值），使 S 最小。（以上所有数据皆为正整数）

$N \le 10^4$，$M \le 20$。

Solution

状态表示方法：记 search(v,floor,R,H,S) 为当前体积为 v，层数为 floor，上一层的 R_i, H_i 为 R, H，已经使用的表面积为 S 的状态。

Solution

状态表示方法：记 search(v,floor,R,H,S) 为当前体积为 v，层数为 floor，上一层的 R_i, H_i 为 R, H，已经使用的表面积为 S 的状态。

可行性剪枝：

- 剩下的若干层都放最大的圆柱，体积也 $< N$。
- 剩下的若干层都放最小的圆柱，体积也 $> N$。

Solution

状态表示方法：记 search(v,floor,R,H,S) 为当前体积为 v，层数为 floor，上一层的 R_i, H_i 为 R, H，已经使用的表面积为 S 的状态。

可行性剪枝：

- 剩下的若干层都放最大的圆柱，体积也 $< N$。
- 剩下的若干层都放最小的圆柱，体积也 $> N$。

最优性剪枝：

- 剩下的若干层都放最小的圆柱，得出的表面积比当前最优解劣。
- 剩下的体积所需的最小表面积加上当前表面积比当前最优解劣。

Solution

状态表示方法：记 search(v,floor,R,H,S) 为当前体积为 v，层数为 floor，上一层的 R_i, H_i 为 R, H，已经使用的表面积为 S 的状态。

可行性剪枝：

- 剩下的若干层都放最大的圆柱，体积也 $< N$。
- 剩下的若干层都放最小的圆柱，体积也 $> N$。

最优性剪枝：

- 剩下的若干层都放最小的圆柱，得出的表面积比当前最优解劣。
- 剩下的体积所需的最小表面积加上当前表面积比当前最优解劣。

其中“剩下若干层都放最小/大的圆柱所用的表面积/体积”和“剩下体积所需的最小表面积”都可以预处理。

Description

有 n 个木块排成一行，从左到右依次编号为 $1 \sim n$。

你有 k 种颜色的油漆，其中第 i 种颜色的油漆足够涂 c_i 个木块。所有油漆刚好足够涂满所有木块，即 $c_1 + c_2 + \ldots + c_k = n$。

相邻两个木块涂相同色显得很难看，所以你希望统计任意两个相邻木块颜色不同的着色方案。

$k \leq 15, c_i \leq 5$。

Solution

search(a, b, c, d, e, l) 表示当前有能涂 1 次的油漆有 a 个，能涂 2 次的有 b 个（c、d、e 类似），前一个颜色还可以涂 l 的方案数，记忆化即可。

Description

将一个 $a*b$ 的数字矩阵进行如下分割：将原矩阵沿某一条直线分割成两个矩阵，再将生成的两个矩阵继续如此分割（当然也可以只分割其中的一个），这样分割了 $n-1$ 次后，原矩阵被分割成了 n 个矩阵。

原矩阵中每一位置上有一个分值，一个矩阵的总分为其所含各位置上分值之和。

对给出的矩阵及 n，求出各矩阵总分的标准差的最小值。

$1 < a, b, n \leq 10$

Solution

标准差 $\sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}$

Solution

标准差 $\sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i-\overline{x})^2}{n}}$

由于 n 和 $\overline{x}$ 不变，那么问题转化为最小化 $\sum_{i=1}^{n}(x_i - \overline{x})^2$。

f[a][b][c][d][k] 表示矩阵 (a,b,c,d) 分成 k 块，最小的 $(x_i - \overline{x})^2$ 之和。记忆化即可。

Description

有一个 5×5 的棋盘，上面有一些格子被染成了黑色，其他的格子都是白色。

对棋盘一些格子进行染色，使得所有的黑色格子能四连通，并且你染色的格子数目要最少。

Solution

答案不大，直接从小到大枚举染色次数，暴力搜索判断。

Description

给一张图，求最大独立集。

Brute Force

最朴素的搜索算法非常简单：用深度优先搜索枚举 V 的子集 $I \in V$，即按一定顺序枚举每个点 $v \in V$ 是否属于 I，一旦存在 $(u, v) \in E$ 使得 $u, v \in I$，就回溯。该算法的复杂度为 $O(2^n * m)$。

Brute Force

最朴素的搜索算法非常简单：用深度优先搜索枚举 V 的子集 $I \in V$，即按一定顺序枚举每个点 $v \in V$ 是否属于 I，一旦存在 $(u, v) \in E$ 使得 $u, v \in I$，就回溯。该算法的复杂度为 $O(2^n * m)$。

一些剪枝，例如：

- 若 $deg(v) = 0$，则不存在与 v 关联的边，故总可以令 $v \in I$。

Brute Force

最朴素的搜索算法非常简单：用深度优先搜索枚举 V 的子集 $I \in V$，即按一定顺序枚举每个点 $v \in V$ 是否属于 I，一旦存在 $(u, v) \in E$ 使得 $u, v \in I$，就回溯。该算法的复杂度为 $O(2^n * m)$。

一些剪枝，例如：

- 若 $deg(v) = 0$，则不存在与 v 关联的边，故总可以令 $v \in I$。
- 若 $deg(v) = 1$，考虑唯一的与 v 关联的结点 u，若 $u \notin I$，则总可以令 $v \in I$；否则，从 I 中删去 u 并加入 v，I 的大小不变。因此总可以令 $v \in I$。

Brute Force

最朴素的搜索算法非常简单：用深度优先搜索枚举 V 的子集 $I\in V$，即按一定顺序枚举每个点 $v\in V$ 是否属于 I，一旦存在 $(u,v)\in E$ 使得 $u,v\in I$，就回溯。该算法的复杂度为 $O(2^n*m)$。

一些剪枝，例如：

- 若 $deg(v)=0$，则不存在与 v 关联的边，故总可以令 $v\in I$。
- 若 $deg(v)=1$，考虑唯一的与 v 关联的结点 u，若 $u\notin I$，则总可以令 $v\in I$；否则，从 I 中删去 u 并加入 v，I 的大小不变。因此总可以令 $v\in I$。
- 搜索时记录当前搜到的独立集的大小的最大值 a，记 P 为 $V-I$ 中不与 I 中结点相邻的点集，当 $|I|+|P|\leq a$ 时可进行最优性剪枝。

将 n 分为两半，大小为 $\frac{n}{2}$。

将 n 分为两半，大小为 $\frac{n}{2}$。

先搜索前半部分，搜索完毕后统计 a[x]，表示在前半部分中选取状态为 x 的子集的最大独立集大小。

将 n 分为两半，大小为 $\frac{n}{2}$。

先搜索前半部分，搜索完毕后统计 a[x]，表示在前半部分中选取状态为 x 的子集的最大独立集大小。

再搜索后半部分，搜索到叶子状态时通过 a 数组来统计答案。

将 n 分为两半，大小为 $\frac{n}{2}$。

先搜索前半部分，搜索完毕后统计 a[x]，表示在前半部分中选取状态为 x 的子集的最大独立集大小。

再搜索后半部分，搜索到叶子状态时通过 a 数组来统计答案。

$O(n * 2^{\frac{n}{2}})$

func(R, P, X)：获取所有包含 R 中结点、可能包含 P 中结点、不包含 X 中结点的极大独立集。其中 $P \cup X$ 是 $V - R$ 中不与 R 中结点相邻的结点集合。调用 func$(\varnothing, V, \varnothing)$ 获取 G 的所有极大独立集。

如果 $P = X = \varnothing$，那么 R 是一个极大独立集。

选择一个 $u \in P \cup X$，使得 $\mid P \cap (\{u\} \cup N(u)) \mid$ 最小。

枚举 $P \cap (\{u\} \cup N(u))$ 中的 v:

- 调用 func$(R \cup \{v\}, P - (\{v\} \cup N(v)), X - (\{v\} \cup N(v)))$
- 然后 $P \leftarrow P - \{v\}$，$X \leftarrow X \cup \{v\}$。

---

func(R, P, X)：获取所有包含 R 中结点、可能包含 P 中结点、不包含 X 中结点的极大独立集。其中 $P \cup X$ 是 $V - R$ 中不与 R 中结点相邻的结点集合。调用 func$(\varnothing, V, \varnothing)$ 获取 G 的所有极大独立集。

如果 $P = X = \varnothing$，那么 R 是一个极大独立集。

选择一个 $u \in P \cup X$，使得 $|P \cap (\{u\} \cup N(u))|$ 最小。

枚举 $P \cap (\{u\} \cup N(u))$ 中的 v:

- 调用 func$(R \cup \{v\}, P - (\{v\} \cup N(v)), X - (\{v\} \cup N(v)))$
- 然后 $P \leftarrow P - \{v\}$，$X \leftarrow X \cup \{v\}$。

复杂度 $O(3^{\frac{n}{3}})$，但随机情况下，实际复杂度远小于这个。~~不过反正应该也用不上~~