搜索算法及其应用

Wearry

Sep 22, 2017

今天主要的讲课内容是搜索算法的应用，主要是提到一些提高搜索效率的一些技巧。

这部分内容在整个信息学竞赛中都属于比较玄学的内容，相信听课过程也会非常的愉悦。

今天主要的讲课内容是搜索算法的应用，主要是提到一些提高搜索效率的一些技巧。

这部分内容在整个信息学竞赛中都属于比较玄学的内容，相信听课过程也会非常的愉悦。

以下是今天的讲课主要内容：

- 启发式搜索
- 广义剪枝
- 随机化搜索的简要介绍

启发式搜索又称为有信息搜索，它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的，这种利用启发信息的搜索过程称为启发式搜索。

启发式搜索中应用最为广泛的搜索技巧当属 $A*$ 和 $IDA*$ 了，前者是 BFS 的启发式版本，后者是前者的迭代加深版本。

$A*$ 和 $IDA*$ 算法中，对每个状态 x 引入一个估价函数 $f(x) = g(x) + h(x)$，其中 $g(x)$ 是目前状态的实际代价，而 $h(x)$ 是目前状态到目标状态的估计代价。

在 $A*$ 算法中，估价函数的作用是调整搜索顺序。而在 $IDA*$ 中，估价函数作为最优性剪枝出现。

$A*$ 和 $IDA*$ 算法中，对每个状态 x 引入一个估价函数 $f(x) = g(x) + h(x)$，其中 $g(x)$ 是目前状态的实际代价，而 $h(x)$ 是目前状态到目标状态的估计代价。

在 $A*$ 算法中，估价函数的作用是调整搜索顺序。而在 $IDA*$ 中，估价函数作为最优性剪枝出现。

显然，为了保证搜索结果的正确性，$h(x)$ 不能大于当前状态到目标状态的最优值。

特殊地，当 $h(x) = 0$ 时，$A*$ 和 $IDA*$ 算法退化为一般的 BFS 算法与迭代加深搜索算法。
这个内容说起来挺复杂，但实际上还比较简单，直接看几道例题吧。

Uva 11163

有 n 只豹子要排队，位置为 $1..n(n\%4=0)$,，其中编号为 1 的为豹王，只有豹王可以和别的豹子交换位置，豹王跳跃有限制，设当前豹王位置为 i:,

如果 $(i\%4=1)$, 那么他可以跳到 $(i+1)$，$(i+3)$，$(i-4)$，$(i+4)$

如果 $(i\%4=2)$, 那么他可以跳到 $(i+1)$，$(i-1)$，$(i-4)$，$(i+4)$

如果 $(i\%4=3)$, 那么他可以跳到 $(i+1)$，$(i-1)$，$(i-4)$，$(i+4)$

如果 $(i\%4=0)$, 那么他可以跳到 $(i-3)$，$(i-1)$，$(i-4)$，$(i+4)$

求豹王最少要跳几次才能排完序，$n \leq 40$

Uva 11163

将豹子按照 $x*4$ 的格子排列，发现每个节点只能走到相邻的节点。

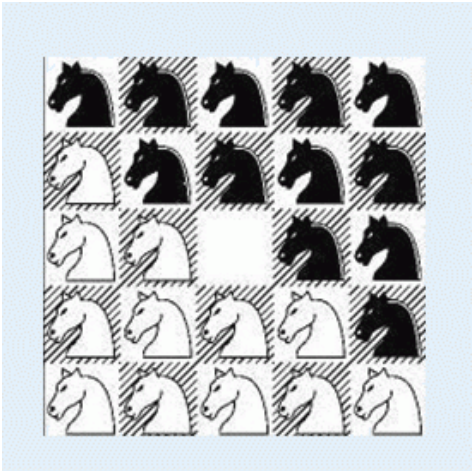
与 8 数码问题相似，估价函数为 $h(s)=\sum_{i=2} d_{i,pos_i}$

d_{i,pos_i} 表示 i 到 pos_i 在 $x*4$ 网格上的曼哈顿距离。

骑士精神

在一个 5×5 的棋盘上有 12 个白色的骑士和 12 个黑色的骑士和一个空位。

给定一个初始的棋盘，怎样才能经过移动变成如下目标棋盘：为了体现出骑士精神，他们必须以最少的步数完成任务。



骑士精神

这道题目与上一题做法差不多，只是每次走的方式发生了变化。

显然这样子的估价函数就是每个骑士到达离自己最近的一个合法位置的最短步数和，这个可以用 dp 预处理出来。

而且原题好像需要满足步数小于 15。

事实上，启发式搜索是一个比较玄学的东西，效果与估价函数关系密切。

一般是在一道题目没有什么正常的思路或者本身就比较玄学的时候尝试的算法。

DFS 的本质就是在一个状态图上进行深度优先遍历，而剪枝过程相当于不去遍历那些显然不可能成为答案的状态。

这里我把所有使得搜索过程中遍历到的状态与转移的数量尽可能少的方法都归于剪枝，一起与大家介绍。

DFS 的本质就是在一个状态图上进行深度优先遍历，而剪枝过程相当于不去遍历那些显然不可能成为答案的状态。

这里我把所有使得搜索过程中遍历到的状态与转移的数量尽可能少的方法都归于剪枝，一起与大家介绍。

事实上，剪枝是搜索所有内容的精髓，几乎不存在套路一说，是考察选手对题目性质分析是否深入的一个较好的标准。

这是今天要提到的主要剪枝策略:

- 结点排序搜索：通过调整当前层的拓展顺序，增强更深层的约束。
- 最优性/可行性剪枝：前者可以配合结点排序搜索使用，后者可以配合估价函数使用。
- 双向搜索：从起点和终点分别向中间搜索或者将问题划分为规模减半的部分解决的方法。
- 迭代加深搜索：对于状态空间无限的问题，通过逐步增大搜索状态的限制层数求解的方法。

这是今天要提到的主要剪枝策略:

- 结点排序搜索：通过调整当前层的拓展顺序，增强更深层的约束。
- 最优性/可行性剪枝：前者可以配合结点排序搜索使用，后者可以配合估价函数使用。
- 双向搜索：从起点和终点分别向中间搜索或者将问题划分为规模减半的部分解决的方法。
- 迭代加深搜索：对于状态空间无限的问题，通过逐步增大搜索状态的限制层数求解的方法。

当然，这里提供的方法和思路还需要结合具体的题目具体分析，下面来看几道例题。

ZOJ1937 Addition chains

构造一个数列 $\{A_i\}$, **要求** $a_1 = 1$ $a_{i(i>1)} = a_j + a_k$ $(j, k < i)$, **并且** $a_t = n$ **且** t **尽量小**.

$n \leq 400$

ZOJ1937 Addition chains

不难想到迭代加深，即枚举 t 的大小。

显然一个最优的非递增序列可以转为递增的最优序列，所以可以限制数列递增。

另一个优化是从大到小枚举 a_j 和 a_k，增加对后面搜索的元素的限制来提高搜索的速度。

ARC 075 F Mirrored

给定正整数 D，求满足 $rev(N) - N = D$ 的 N 的数量，其中 $rev(N)$ 是 N 将十进制下的数字反转所得的数。$rev(123) = 321, rev(4000) = 4$

$D \leq 10^9$

ARC 075 F Mirrored

显然我们只需考虑 10^{18} 以内的 N。
考虑枚举 N 的位数，然后从低位开始填数，进行如下可行性剪枝：

若当前差值加上最大/最小的可能的值，仍然小于/大于 D，则返回。

因为先决策的是差值较大的位置的数字，所以对后面的约束会比较强，就能较快地搜出答案。

Rikka with Sequence II

给出一个长度为 n 的数列，要求从中选出一个非空的子集，满足选出的数的平均数小于等于中位数。求方案数。

$n \leq 40, A_i \leq 10^9$

Rikka with Sequence II

发现中位数的处理比较棘手，因为无法快速合并。

注意到中位数一定是数列中某两个数的平均数，接着可以枚举中位数。

Rikka with Sequence II

发现中位数的处理比较棘手，因为无法快速合并。

注意到中位数一定是数列中某两个数的平均数，接着可以枚举中位数。设这两个数是 l 和 r，然后给数列中所有数减去中位数，那么这时的方案就相当于从 $[1, l)$ 和 $(r, n]$ 中选出同等数量的数，使得它们的和 ≤ 0 。

Rikka with Sequence II

发现中位数的处理比较棘手，因为无法快速合并。

注意到中位数一定是数列中某两个数的平均数，接着可以枚举中位数。设这两个数是 l 和 r，然后给数列中所有数减去中位数，那么这时的方案就相当于从 $[1, l)$ 和 $(r, n]$ 中选出同等数量的数，使得它们的和 ≤ 0 。

如果一个数原来的下标 $< l$ 则权重为 1，否则为 -1。
则问题转化为给出若干数，求选出一些数使得权重和为 0，且权值和 ≤ 0 的方案数。

Rikka with Sequence II

经典问题

使用 $Meet\ in\ the\ middle$ 的技巧优化。
将数列分成均等的两个部分，分别处理出两边的所有情况，然后按权重分组，每组内排序统计答案即可。

Rikka with Sequence II

经典问题

使用 $Meet\ in\ the\ middle$ 的技巧优化。
将数列分成均等的两个部分，分别处理出两边的所有情况，然后按权重分组，每组内排序统计答案即可。

复杂度 $O(2^{n/2}n^2)$

上述做法虽然是正确的，但如果常数处理不好还是会超时，有没有更好的做法呢？

Rikka with Sequence II

发现其实没有必要排序。

由于每一次我们都是选择一个数，考虑其加或不加。

对于加或者不加的两种情况依然是有序的，所以可以在搜索的时候归并，保证每次搜完后都有序即可，可以优化掉一个 n 的复杂度。

随机化搜索的原形应该是二分搜索或者三分搜索，即求取值范围内函数的最低或者最高点。

随机化搜索的原形应该是二分搜索或者三分搜索，即求取值范围内函数的最低或者最高点。

然而二分或者三分算法面对以下两种情况却无能为力：

- 当这样的函数图像不是一个简单的单峰函数时。
- 当这样的函数自变量不再是一个数，而是一个向量或者一个集合这类高维情形时。

随机化搜索的原形应该是二分搜索或者三分搜索，即求取值范围内函数的最低或者最高点。

然而二分或者三分算法面对以下两种情况却无能为力：

- 当这样的函数图像不是一个简单的单峰函数时。
- 当这样的函数自变量不再是一个数，而是一个向量或者一个集合这类高维情形时。

随机化算法的问题中常见的处理方式是将原问题的最优化条件转化成一个函数，然后通过随机化的过程不断调整当前函数的自变量取值（这个过程往往要考虑原问题的函数图像的一些性质来使用合适的方法），达到找到一定范围和精度要求下的最优答案的目的。

这里由于讲课人水平有限，给大家几个链接参考一些常见的随机化算法：

- 模拟退火
- 粒子群算法
- 遗传算法

事实上，在实际应用的大多数情况中，模拟退火算法都足够优秀，并且很好写。所以我们讲的例题也都是用模拟退火来做的。

Bzoj 3680 吊打 XXX

给定平面内的 N 个点，求一个点，使得它到其他点的距离与重力的乘积和最小。

$N \leq 10000$

Bzoj 3680 吊打 XXX

模拟退火的简单应用。

根据当前这个点计算出来的值随机化调整这个要求的点，注意几个提高精度的小技巧：选择一个接近最优解的点作为初始点，最后再答案附近寻找更优解。

Bzoj 3754 最小方差生成树

给你一个 N 个点，M 条边的无向联通图，求它的一个生成树，使得这棵数的所有边权值的方差最小。

$N \leq 100, M \leq 2000$

Bzoj 3754 最小方差生成树

在这个问题中我们要最优化值的是一个边集的方差，同时这个边集需要满足是原图的一个生成树，直接随机搜索这个边的集合显然不是一个优秀的方法。

Bzoj 3754 最小方差生成树

在这个问题中我们要最优化值的是一个边集的方差，同时这个边集需要满足是原图的一个生成树，直接随机搜索这个边的集合显然不是一个优秀的方法。

这时就需要分析题目的性质了，发现对于函数

$$\sum_{i=1}^{n}(a_i - \bar{a})^2$$

如果将 $\bar{a}$ 看成是自变量 x, 最低点显然是在 $x = \bar{a}$ 时取到的。

Bzoj 3754 最小方差生成树

在这个问题中我们要最优化值的是一个边集的方差，同时这个边集需要满足是原图的一个生成树，直接随机搜索这个边的集合显然不是一个优秀的方法。

这时就需要分析题目的性质了，发现对于函数

$$\sum_{i=1}^{n}(a_i-\bar{a})^2$$

如果将 $\bar{a}$ 看成是自变量 x，最低点显然是在 $x=\bar{a}$ 时取到的。

所以我们可以随机搜索这个 $\bar{a}$。然后将边权赋为 $(w_i-\bar{a})^2$ 做最小生成树。根据刚才的分析我们知道，这样算出来的答案一定大于等于实际上的方差，并且当 $\bar{a}$ 与边权的平均值恰好相等时，会与实际的方差相等。

谢谢大家