顶点覆盖问题

By 麦骏 阮炜霖 柯瑞凯

目录页

目录
CONTENTS

什么是顶点覆盖问题

最小顶点覆盖:选择图中最少的顶点来覆盖所有的边。

最小权顶点覆盖:给定点权（即代价），求权值和最小的顶点覆盖方案。

样例

输入:

7 7

1 100 1 1 1 100 10

1 6

2 4

2 5

3 6

4 5

4 6

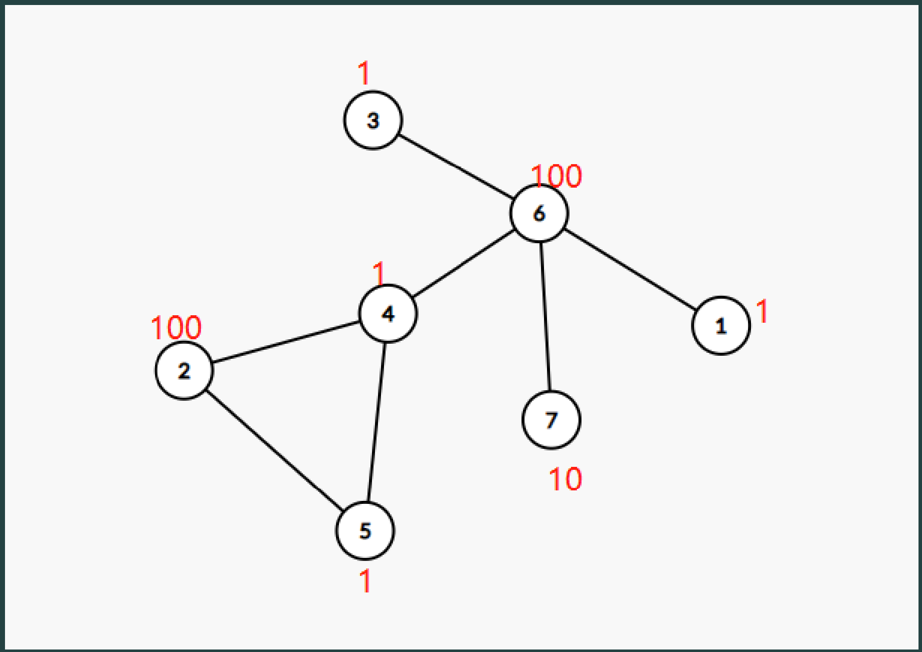
6 7

输出:

14

1011101

作业的数据➔



爆搜！

考虑每个点只有选和不选的情况，那么相当于是2^n规模的决策方案，每种方案的计算权值O(n)，检查边是否完全覆盖O(e)，总共复杂度就是O(e*2^n)了(你就当边比点多嘛)

课本上的样例只有7个点，学过搜索~~就可以拿去写报告了~~

就可以考虑回溯和剪枝等策略。

爆搜是人类智慧的结晶！赞美！

爆搜！

为了拿分让程序跑得更快，在这基础上想想怎么优化？

Bitset预处理每个点表示的边集：可以降低检查的复杂度

限界剪枝：如果把排列写成dfs的形式时，多写一句超过答案后返回。

爆搜！

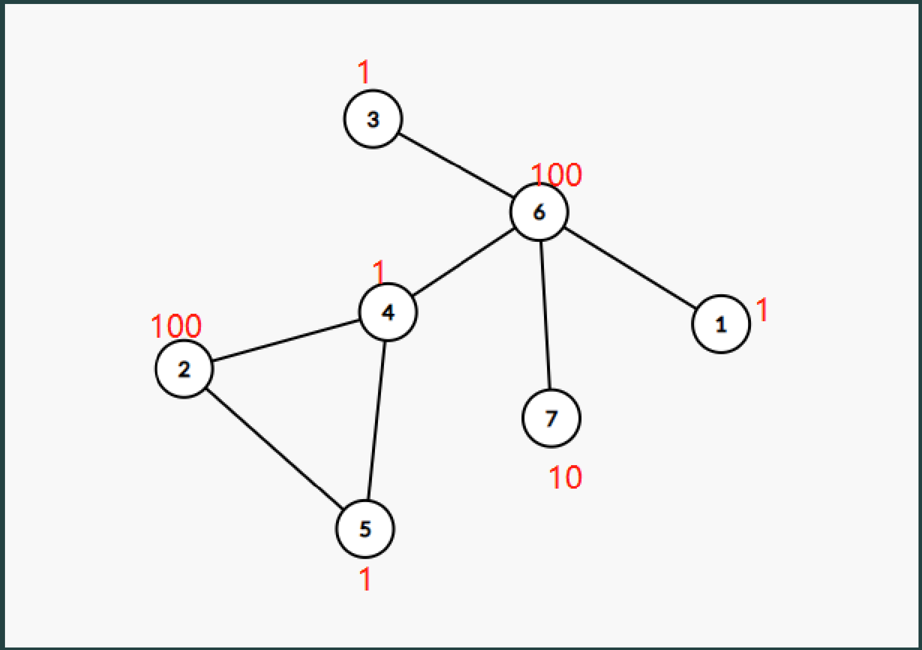
```cpp
void pre(int stp,int sum,int cov){ //遍历点数，点权值和,覆盖的边数
    //cout<<stp<<" "<<sum<<" "<<cov<<" "<<"\n";
    if(sum>=ans) return; //最优性剪枝
    if(stp==n+1){
        if(cov==m){
            ans=sum;
            for(int i=1;i<=n;++i) cv[i]=chs[i];
        }
        return;
    }
    chs[stp]=0;
    pre(stp+1,sum,cov); //不选该节点
    for(int i=head[stp];i;i=e[i].nxt){
        if(!vis[i]) ++cov;
        ++vis[i];
        ++vis[i^1];
    }
    chs[stp]=1;
    pre(stp+1,sum+val[stp],cov);
    for(int i=head[stp];i;i=e[i].nxt){
        --vis[i];
        --vis[i^1];
        if(!vis[i]) --cov;
    }
}
```

←来自阮炜霖的俗手

复杂度更低的解法

贪心?

作业的数据→

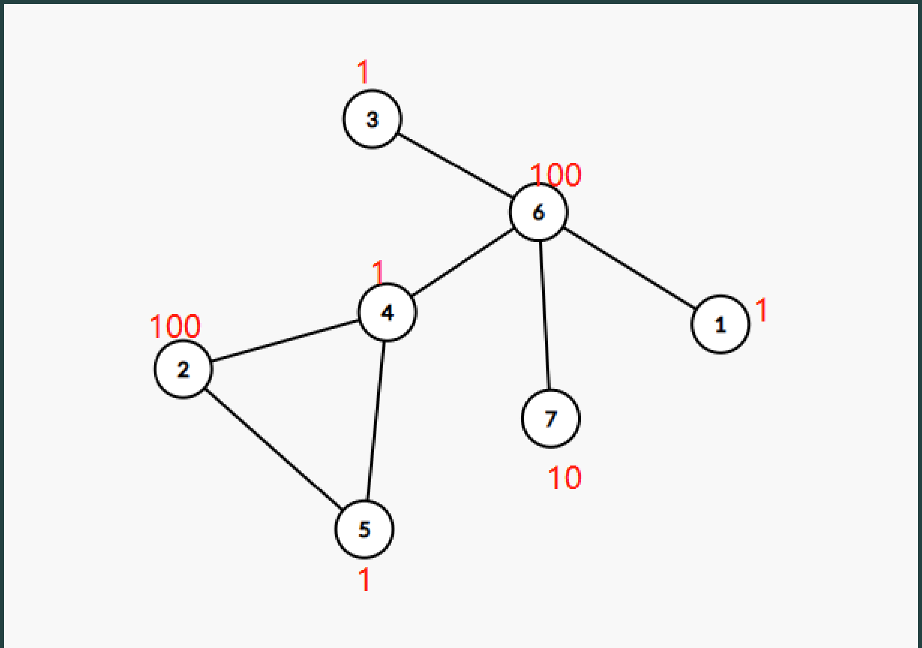


复杂度更低的解法

贪心? ×

如果我只考虑度数再看权值，那我可能会按6->4->5那样选，代价高达102。
如果只考虑代价也同样会犯错误。
（大家类比一下背包问题，这里不多赘述）
我们如何同时考量每个点的代价和收益。

作业的数据→



分支限界法

引用自OI-wiki

启发式搜索

本页面将简要介绍启发式搜索及其用法。

简介

启发式搜索（英文：heuristic search）是一种改进的搜索算法。它在普通搜索算法的基础上引入了启发式函数，该函数的作用是基于已有的信息对搜索的每一个分支选择都做估价，进而选择分支。简单来说，启发式搜索就是对取和不取都做分析，从中选取更优解或删去无效解。

A*

本页面将简要介绍 A*算法。

简介

A*搜索算法（英文：A*search algorithm，A*读作 A-star），简称 A*算法，是一种在图形平面上，对于有多个节点的路径求出最低通过成本的算法。它属于图遍历（英文：Graph traversal）和最佳优先搜索算法（英文：Best-first search），亦是 BFS 的改进。

定义起点 s，终点 t，从起点（初始状态）开始的距离函数 $g(x)$，到终点（最终状态）的距离函数 $h(x)$，$h^*(x)$[1]，以及每个点的估价函数 $f(x)=g(x)+h(x)$。

A*算法每次从优先队列中取出一个 f 最小的元素，然后更新相邻的状态。

如果 $h \leq h*$，则 A*算法能找到最优解。

上述条件下，如果 h 满足三角形不等式，则 A*算法不会将重复结点加入队列。

当 $h=0$ 时，A*算法变为 Dijkstra；当 $h=0$ 并且边权为 1 时变为 BFS。

引用自OI-wiki

这跟今天要讲的分支限界法有啥关系?

分支限界法不单单只是一个简单的BFS，名副其实它需要通过紧缩上下界限来找到更优的答案。我们在**电路板排列**问题中，有一种解法是利用曼哈顿距离计算下一个点到答案的**估价函数**，从而更快得到我们想要的路径。在最小权顶点覆盖问题中，我们思考如何设计估价函数，可以寻找下一个最划算的顶点。

```cpp
#include<bits/stdc++.h>
#define rep(i,x,y) for(int i=x; i<=y; ++i)
using namespace std;
const int N=55;
int n,m,a[N],ans;
bitset <N> mp[N],bit[N];
struct D {
    D() { x=f=0,G(),H(); }
    bitset <N> vis;
    int x,f,g,h;
    bool check() {
        bitset <N> tmp=(vis|mp[x])&bit[x];
        return tmp==vis;
    }
    void getans() {
        rep(i,x+1,n) {
            bitset <N> tmp=(vis|mp[i])&bit[x];
            if(tmp!=vis) f+=a[i],vis[i]=1;
            ++x;
        }
        G(),H();
    }
    void G() {
        g=0;
        rep(i,x+1,n) {
            bitset <N> tmp=(vis|mp[i])&bit[x];
            if(tmp!=vis) g+=a[i];
        }
    }
    void H() {
        h=f;
        rep(i,x+1,n) {
            bitset <N> tmp=(vis|mp[i]);
            if(tmp!=vis) h+=a[i],vis[i]=1;
        }
        vis&=bit[x];
    }
} ;
```

```cpp
int getint() {
    char ch;
    while(!isdigit(ch=getchar()));
    int x=ch-48;
    while(isdigit(ch=getchar())) x=x*10+ch-48;
    return x;
}
int main() {
    n=getint(),m=getint();
    rep(i,1,n) a[i]=getint();
    rep(i,1,n) rep(j,1,i) bit[i][j]=1;
    rep(i,1,m) {
        int u=getint(),v=getint();
        mp[u][v]=mp[v][u]=1;
    }
    D x;
    q.push(x),ans=x.h;
    while(!q.empty()) {
        x=q.top(),q.pop();
        if(x.f+x.g==x.h) break;
        int p=x.x+1;
        ++x.x,x.G(),x.H();
        if(x.f+x.g<=ans && x.check()) {
            ans=min(ans,x.h);
            q.push(x);
        }
        x.vis[p]=1,x.f+=a[p],x.G(),x.H();
        if(x.f+x.g<=ans) {
            ans=min(ans,x.h);
            q.push(x);
        }
    }
    printf("%d\n",x.h);
    x.getans();
    rep(i,1,n) putchar('0'+x.vis[i]);
    puts("");
    return 0;
}
```

←来自麦骏的妙手

这里只能大致介绍

目录页

Next 如何写分支限界?

krk来教你

分支限界法

柯瑞凯

6月8号

02

最大权独立集

01

最小权顶点覆盖

03

最大团

分支限界法

01

最小权顶点覆盖

1. 使用小根堆来维护分支限界法中的状态节点先后扩展次序
2. 将当前已经获得的权值和作为优先队列内排序的关键字
3. 每次将优先队列的队首元素进行判定，达到叶子节点即为答案
4. 每次对当前状态的下一个点选取时，均需要进行限界剪枝
5. 每次对当前状态的下一个点不选时，均需要进行约束剪枝

约束剪枝：若不选接下来的点，将剩下点假设全部加入答案中查看是否能达到顶点覆盖的条件。

限界剪枝：若选择接下来的点，判断该点加入时是否产生新的边覆盖。

注意可以将所有点的权值从大到小排序后的次序在来进行计算

02

最大权独立集

1 换种思维方式

2 考虑寻找一个最大权独立集

3 选取的点之间互不相邻（无直接连边）

4 剩下的点即构成最小点权覆盖

5 用分支限界法解决

采用大根堆维护 20%

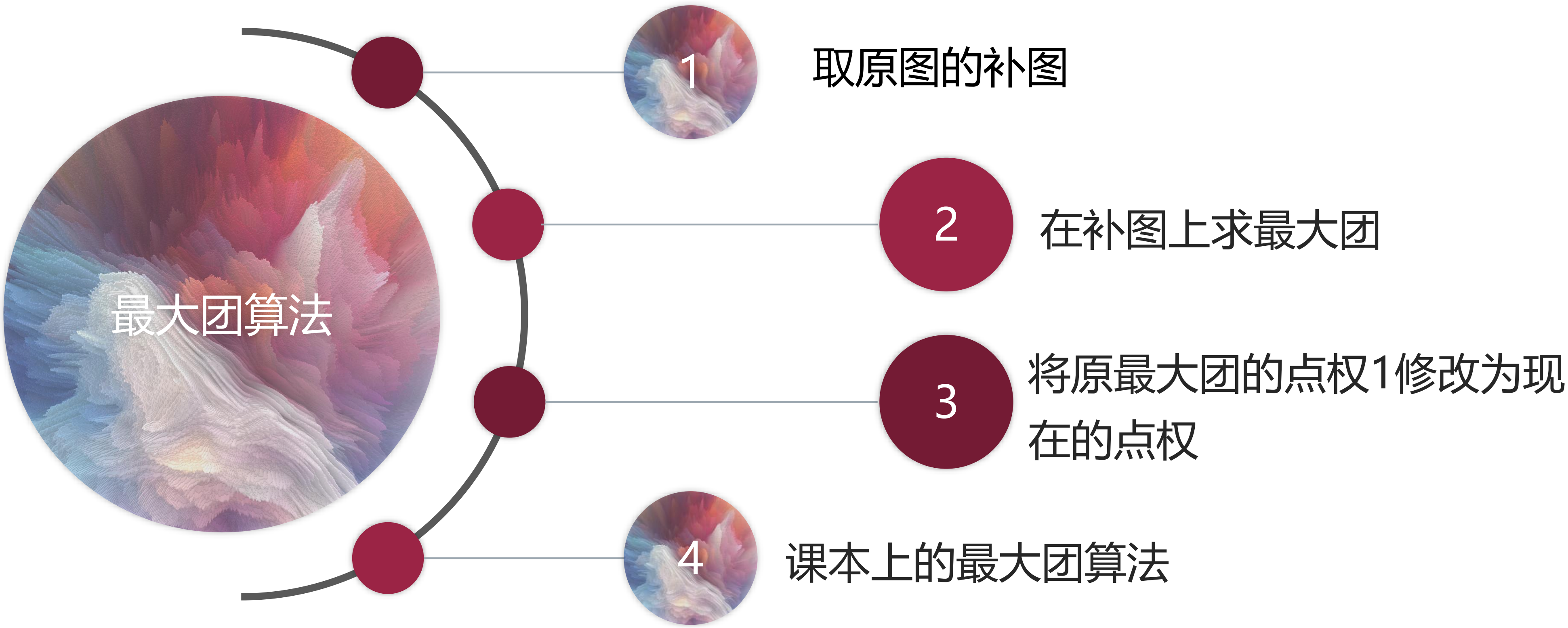
将当前以获得权值与剩下点的权值和作为关键字 50%

同样对点选与不选进行操作 70%

约束剪枝：选入的点不与当前答案集合的点连边 100%

03

最大团算法


最大团算法
1
取原图的补图
2
在补图上求最大团
3
将原最大团的点权1修改为现在的点权
4
课本上的最大团算法

最小权顶点覆盖

MJ总结

分支限界法总结

课堂展示

麦骏

信息科学技术学院

2022 年 6 月 8 日

分支限界法

- 搜索算法主要优化在于剪枝力度。
- 分支限界法的精髓在于搜索树子树内上下界函数的确定。
- 当加入优先队列优化时，每次会选择一个子树可能最优的节点进行扩展。
- 但上下界相等时，说明已经获得子树内最优的答案。又因为每次选择的节点为全局可能最优的节点。那么当上下界相等时，该节点就是全局最优节点。
- 若需要求最小值，那么子树下界作为优先级，子树上界用来更新答案。若需要求最大值，则子树上界作为优先级，下界用来更新答案。

迭代加深搜索

- 有一个长度为 n 全排列 p_i. 每次可以将某个前缀 reverse, 求最少几次操作可以将排列升序排序.
- $n \leq 25$.

迭代加深搜索

- 若采用分支限界法，则子树下界可以定义为 $|p_i - p_{i+1}| \neq 1$ 的位置的个数，而子树上界可以贪心求得。
- 子树上界收敛速度慢，时空的浪费严重。

迭代加深搜索

- 观察到答案与子树上界都是 $O(n)$，可以从小到大枚举答案进行 dfs 搜索。
- 每次剪枝剪去子树下界大于枚举的答案的节点。

总结

- 以上算法主要通过对于子树上下界估值进行剪枝，上下界确定的精确度决定了剪枝的力度。
- 另外，当上下界拥有较为特殊的性质时，可以调整搜索方式，进行优化。

Thanks!