
算法分析与设计

实验 06

实验目的

- 掌握分支限界法的解题步骤。
- 掌握数值随机化算法、舍伍德算法和拉斯维加斯算法。

实训内容

1、算法设计题（完善最大团问题，书本例题）；

● 思路

设状态为所有已经加入团的节点按照编号（0~n-1）状态压缩后的结果，上界为所有还未试图加入过团的节点数+已加入团的节点数

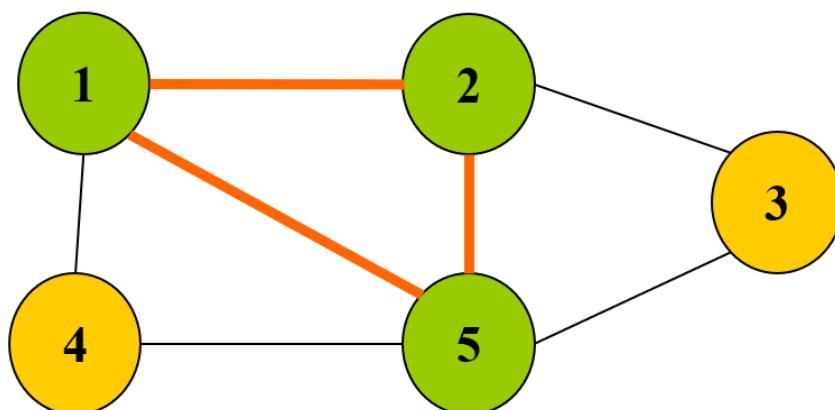
每次搜索下一个节点时选择 up（上界）最大的，假设为 t，搜索 t 时 t 若为叶子节点尝试更新答案。

若非叶子节点则对于 t 正在考虑是否要加入团的节点 t.x 有两种情况：

1. 不加入，上界减小，加入队列
2. 加入上界不变，检查是否合法，合法加入队列

● 代码

样例：



Node 节点类:

因为是 NP 难问题, 所以 n 不可能太多, 所以直接状态压缩

```
11 int n, m;
12 bool mp[N][N];
13 ll best = -1e9, ans;
14 struct node {
15     //在搜第x个点
16     int x;
17     //当前状态压缩后已经选取的点的情况
18     ll now;
19     //上界
20     int up;
21     //优先队列根据上界比较
22     bool operator <(const node other) const {
23         return up < other.up;
24     }
25 };
26 void bfs() {
```

主程序

```
6 void bfs() {
7     priority_queue<node> q;
8     //初始上界为主集
9     q.push({ 0,0,n });
10    while (q.size()) {
11        auto t = q.top(); q.pop();
12        //最优解情况 上界小于最优
13        if (t.up <= best) continue;
14        //如果选中了节点
15        if (t.x == n) {
16            if (cnt1(t.now) > best) {
17                ans = t.now;
18                best = cnt1(t.now);
19            }
20            continue;
21        }
22        //不选t.x情况
23        node st = t;
24        st.x++; st.up--;
25        q.push(st);
26        //选了t.x的情况: 上界不变,x++,now&上当前位置并判断合法性
27        st = t;
28        st.x++; st.now |= (1 << t.x);
29        bool flag = true;
30        for (int i = 0; i < n; i++) {
31            if ((t.now >> i & 1) && !mp[i][t.x]) {
32                flag = false;
33                break;
34            }
35        }
36        if (flag) q.push(st);
37    }
38 }
39
40 void solve() {
41     cin >> n >> m;
42     for (int i = 0; i < m; i++) {
43         int a, b; cin >> a >> b;
44         a--; b--;
45         mp[a][b] = mp[b][a] = 1;
46     }
47     bfs();
48     cout << "最大团有: " << best << "个节点\n";
49     cout << "分别为: ";
50     int t = 1;
51     while (ans) {
52         if (ans & 1) cout << t << " ";
53         ans >>= 1;
54         t++;
55     }
56 }
```

```
1 Test 1 Passed
2
3 Input 1:
4 5 7
5 1 2
6 1 5
7 2 5
8 3 2
9 3 5
10 4 1
11 4 5
12
13 Expected Output :
14 最大团有: 3个节点
15 分别为: 1 2 5
16
17 Obtained Output :
18 最大团有: 3个节点
19 分别为: 1 2 5
20
21
22
23
```

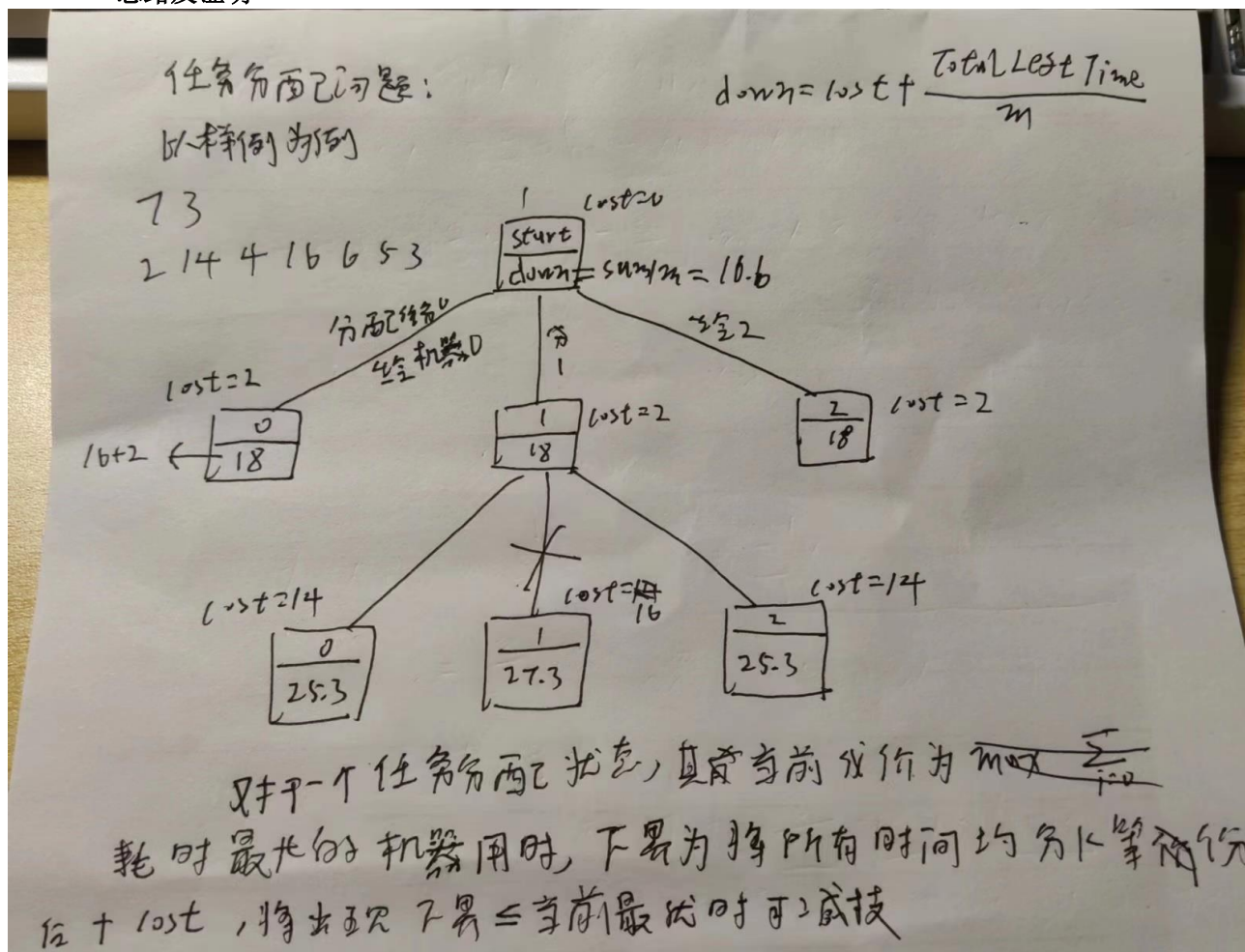
All test cases passed.

来源: CodePal

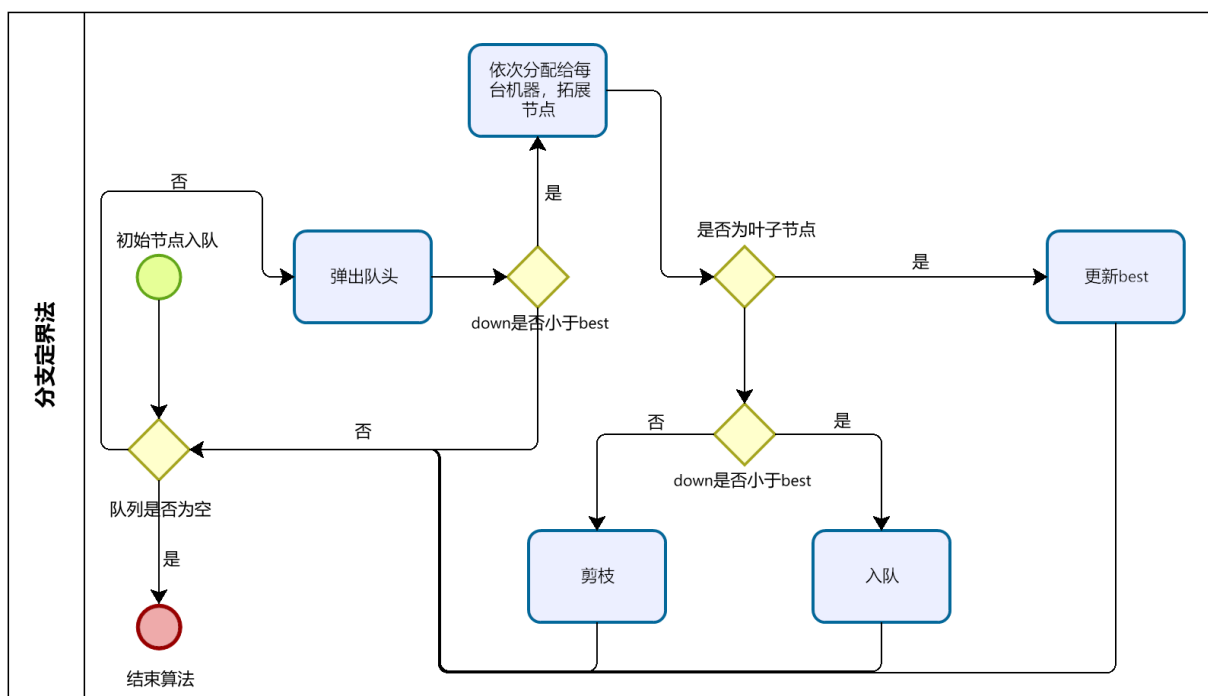
Show Results

2、算法设计题（最佳调度问题，6-8，P201）：使用分支限界法实现最佳调度问题；

● 思路及证明



● 程序流程图



- 代码

Node 节点类:

```
8  ll n,m;
9  vector<int> time;
10 double s;
11 int best=1e9;
12 struct node{
13     //当前在搜索的任务
14     int k;
15     //最大等待时间
16     int cost;
17     //每个机器实际代价
18     vector<int> q;
19     //下界 最大等待时间+剩余任务总时间/m
20     double down;
21     double leftTotalTime;
22     //为了优先队列从小到大, 这里反转比较符
23     bool operator <(const node other) const{
24         return down>other.down;
25     }
26 };
27 node get(node &x,int p){
28     node now=x;
```

主程序:

The screenshot shows a C++ IDE with two main windows. The left window displays the main program code, which includes the `node` struct definition and the `bfs()` and `solve()` functions. The `bfs()` function uses a priority queue to explore different task assignments, updating the best solution found. The `solve()` function reads input and calls `bfs()`. The right window shows the test results, indicating that all test cases passed. The test input is as follows:

```
1 Test 1 Passed
2
3 Input 1:
4 7 3
5 2 14 4 16 6 5 3
6
7 Expected Output :
8 17
9
10 Obtained Output :
11 17
12
13
14
15
```

At the bottom of the IDE, a status bar indicates "All test cases passed." and "来源: CodePal".

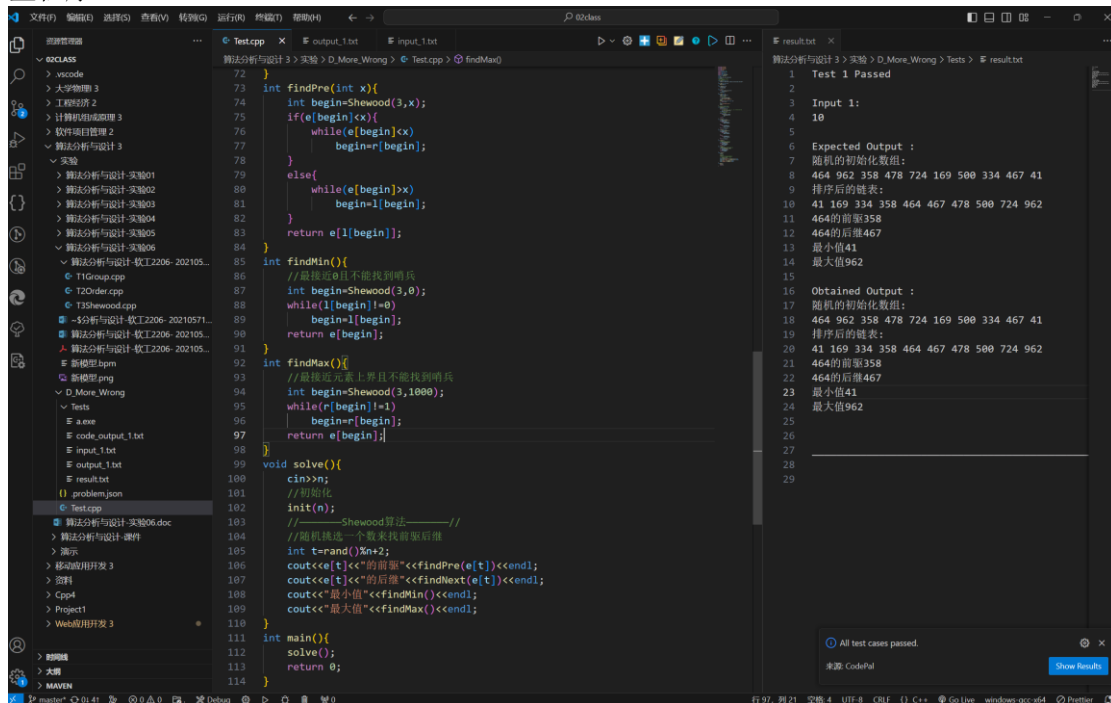
3、算法分析和设计题 (7-7, P237)：有序链表的查找问题。

● 算法流程

通过 Shewood 找到 \sqrt{n} 个位置, 返回最接近 x 的数的下标—>如果大于查找数 x 则向左查到 x , 否则向右查到 x —>返回结果

● 代码 排序复杂度 $O(N\log N)$, 查找复杂度 $O(\log_{\sqrt{N}}(N))$

主程序



```
72 }
73 int findPre(int x){
74     int begin=Shewood(3,x);
75     if(e[begin]<x){
76         while(e[begin]<x)
77             begin=r[begin];
78     }
79     else{
80         while(e[begin]>x)
81             begin=l[begin];
82     }
83     return e[l[begin]];
84 }
85 int findMin(){
86     //最近0且不能找到哨兵
87     int begin=Shewood(3,0);
88     while(l[begin]!=0)
89         begin=l[begin];
90     return e[begin];
91 }
92 int findMax(){
93     //最近元素上界且不能找到哨兵
94     int begin=Shewood(3,1000);
95     while(r[begin]!=1)
96         begin=r[begin];
97     return e[begin];
98 }
99 void solve(){
100     cin>>n;
101     //初始化
102     init(n);
103     //-----Shewood算法-----//
104     //随机挑选一个数来找前驱后继
105     int t=rand()*n+2;
106     cout<<e[t]<<"的前驱"<<findPre(e[t])<<endl;
107     cout<<e[t]<<"的后继"<<findNext(e[t])<<endl;
108     cout<<"最小值"<<findMin()<<endl;
109     cout<<"最大值"<<findMax()<<endl;
110 }
111 int main(){
112     solve();
113     return 0;
114 }
```

构建有序双向链表



```
11 const int N=1e5+10;
12 int n,m;
13 int h=-1,e[N],r[N],l[N],idx=2;
14 int w[N];
15 //将w的x号元素加入k右边
16 //0和1为左右哨兵
17 void addr(int k,int x){
18     e[idx]=x,r[idx]=r[k],l[idx]=k,l[r[k]]=idx,r[k]=idx++;
19 }
20 void addl(int k,int x){
21     addr(l[k],x);
22 }
23 void remove(int k){
24     r[l[k]]=r[k];
25     l[r[k]]=l[k];
26 }
27 void show(){
28     for(int i=r[0];i!=1;i=r[i])
29         cout<<e[i]<<" ";
30     cout<<endl;
31 }
32 void init(int x){
33     w[0]=-inf;w[1]=inf;
34     r[0]=1,l[1]=0;
35     //随机初始化w
36     for(int i=2;i<=x;i++)
37         addr(0,rand()*1000);
38     cout<<"随机的初始化数组:\n";
39     show();
40     sort(e+2,e+x+2,[](int x,int y){
41         return x>y;
42     });
43     cout<<"排序后的链表:\n";
44     show();
45 }
```

寻值函数

```
int Shewood(int x){
    int k=sqrt(n);
    set<int> st;
    while(st.size()<k)
        st.insert(rand()%n+2);
    int now=inf,p=-1;
    for(auto t:st) //找一个最接近的小标返回
        if(abs(e[t]-x)<now){
            now=abs(e[t]-x),p=t;
        }
    return p;
}

int findNext(int x){
    int begin=Shewood(x);
    if(e[begin]<x)
        while(e[begin]<x) begin=r[begin];
    else while(e[begin]>x) begin=l[begin];
    return e[r[begin]];
}

int findPre(int x){
    int begin=Shewood(x);
    if(e[begin]<x)
        while(e[begin]<x) begin=r[begin];
    else while(e[begin]>x) begin=l[begin];
    return e[l[begin]];
}

int findMin(){
    int begin=Shewood(0); //最接近 0 且不能找到哨兵
    while(l[begin]!=0)
        begin=l[begin];
    return e[begin];
}

int findMax(){
    int begin=Shewood(1000); //最接近元素上界且不能找到哨兵
    while(r[begin]!=1)
        begin=r[begin];
    return e[begin];
}
```