搜索小结

DFS

PAT1103有以下几点体会：

1.开销大的函数可以建表

```
void init() {
        int temp = 0, index = 1;
        while (temp <= n) {
                v.push_back(temp);
                temp = pow(index, p);
                index++;
        }
}
```

2.结束的写法

(1)正常分开两种结束，使用**return 0;**

```
if (maxFacSum == -1) {
        printf("Impossible");
        return 0;
}
printf("%d = ", n);
for (int i = 0; i < ans.size(); i++) {
        if (i != 0) printf(" + ");
        printf("%d^%d", ans[i], p);
}
return 0;
```

(2)if else结构

```
        if(!max_sum)
                printf("Impossible");
        else {
                printf("%d = ", n);
                for (i = 0; i < K-1; ++i)
                        printf("%d^%d + ", v[i],p);
                printf("%d^%d", v[K-1], p);
        }
        return 0;
```

3.resize和reverse

- resize()会改变当前容器的“内容”(空间大小和内容值都可被改变) 。该函数有一或两个参数，默认补0。
- reserve()只会改变当前容器的“容量”大小。
- 当resize改变了size后，capacity也可能被改变。但当reserve改变了capacity后，size并不会变化

4.引用自《算法基础实践》

枚举的几个要点

(1)搜索结构

- 递归结构 最后一层--剪枝层(前2顺序可换)--递归层

```
void dfs(int num,int cur,int sum){
    if (num>k) return;
    if(num==k&&sum==n){
        if(larger(temp, ans))
            ans=temp;
    }else if(num<k&&sum<n){
        temp.push_back(cur);
        dfs(num+1, cur, sum+(int)pow(cur, p));
        temp.pop_back();
        if(sum+(k-num)*pow(cur+1, p)<=n)
            dfs(num, cur+1, sum);
    }
}
```

- 搜索结构

```
void dfs(int n,int k,int p,int sum) {
        for (int i = k<K?t[K-k-1]:lim; i>=1; --i) {
                if(k>1) {
                        if(n-k+1>=pow(i,p)) {
                                t[K - k] = i;
                                dfs(n - pow(i, p), k - 1, p, sum + i);
                        }
                }
                if(k==1) {
                        if (n == pow(i, p)) {
                                t[K - 1] = i;
                                sum += i;
                                if(sum>max_sum) {
                                        max_sum = sum;
                                        v=t;
                                }
                                break;
                        } else if (n > pow(i, p))
                                break;
```

```
                }
            }
}
```

(2)搜索的顺序

一般先试选择少的步骤，后试选择多的步骤 eg:七巧板先拼大块

(3)剪枝与状态更新

状态更新:若变量与栈的回退有关，可将其加入形参而不是全局变量

```
sum += i;
```

剪枝:可行性和最优性剪枝

BFS

PAT1091有如下体会

1.push和pop处的安排

visit[i][j][k] = 1;和sum++;的位置选取: 由于此题是多米诺骨牌式访问，若不提前置1会重复计数(不然没有起到搜索的效果)，只要保证了这个前提，sum++放两个地方都行

```
int bfs(int i,int j,int k)
{
    queue<Node> q;
    Node temp;
    int sum=0;
    node = {i,j,k};
    q.push(node);
    visit[i][j][k] = 1;
    while(!q.empty()){
        temp=q.front();
        q.pop();
        sum++;
        for (int r = 0;r<6; ++r){
            node = {temp.i + l[r], temp.j + m[r], temp.k + n[r]};
            if (node.i < 0 || node.j < 0 || node.k < 0 || node.i >= L || node.j >=
M || node.k >= N || visit[node.i][node.j][node.k]||!sq[node.i][node.j][node.k])
                continue;
            else
            {
                q.push(node);
                visit[node.i][node.j][node.k] = 1;
            }
```

```
            }

        }
        if(sum>=T)
            return sum;
        else
            return 0;
    }
```

2.注意**temp**的设置，每次**for(6)**得保证不变