# 数据结构与算法递归与搜索

陈宇琪

2020 年 4 月 12 日

**摘要**

主要内容包括递归、bfs、dfs。
提交方式：除部分编程题在 EOJ 上提交，其他题目在超星上提交。
**ddl：2020-04-12**

## 目录

# 1 知识点补充

## 1.1 搜索与剪枝

### 1.1.1 可行性剪枝

如果当前条件不合法就不再继续搜索，直接 return。

例如：在 n 皇后问题中，如果搜索到当前状态已经出现了冲突，则可以不用继续搜索下去了！

具体而言，对于 4 皇后问题而言，由于每行最多只能放 1 个皇后，如果前 2 行放置的皇后分别在 $(1,1)$ 和 $(2,2)$，则已经发生冲突，可以直接 return。

### 1.1.2 最优性剪枝

如果当前条件所创造出的答案必定比之前的答案大，那么剩下的搜索就毫无必要，甚至可以剪掉。

例如：搜索目标是最小化某个目标函数，如果当前的状态必定导致最后的值大于已经找到的一个局部最优解，则可以直接 return。

### 1.1.3 记忆化搜索

如果对于相同情况下必定答案相同，就可以把这个情况的答案值存储下来，以后再次搜索到这种情况时就可以直接调用。

### 1.1.4 * 启发式搜索

定义：启发式距离 $h(x)$，满足 $\forall x \in X, h(x) \le h^*(x)$，其中 $X$ 为所有状态的集合，$h^*(x)$ 为从 $x$ 到目标状态的实际最优解。

在实际问题中，往往 $h^*(x)$ 是暂时未知的，所以用一个下界函数 $h(x)$ 去近似。

其中 $h(x)$ 为启发式函数，所以启发式搜索一般采用优先级队列完成。

定义：$f(x) = g(x) + h(x)$，其中 $g(x)$ 为从起点到状态 $x$ 的距离（这个在搜索过程中是已知的）。

算法描述：首先将起点 $(s, f(s))$ 加入优先级队列，每次从优先级队列中找到最小的 $f(u)$，更新 $u$ 可以到达的所有状态 $v$，将 $(v, f(v))$ 加入优先级队列，直到 $u$ 为终点结束。

启发式搜索可以找到从起点到终点的最优解，而且一般比直接搜索快。

对于最短路问题，可以假设 $h(x)$ 为从 $x$ 到终点的直线距离。

# 2 选择题

1、一个递归算法必须包括（ ）。

A. 递归部分 B. 终止条件和递归部分 C. 迭代部分 D. 终止条件和迭代部分

# 3 简答题

1、指出下面算法的功能（解决什么问题），分析下面程序的复杂度，并指出是否可以优化这个算法：

Listing 1: **ques1.cpp**

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll f(int x)
```

```
{
        if (x==0) return 1;
        if (x==1) return 1;
        return f(x-1)+f(x-2);
}
int main()
{
        int n;
        cin>>n;
        cout<<f(n)<<endl;
}
```

2、对于一个简单的 BFS 问题，如果一个状态可以延展出 n 个状态，并且我们需要穷举所有 m 步可以到达的状态，那么整个算法复杂度是多少？

## 4 编程题

1、根据辗转相除定义，我们知道 $gcd(a,b) = gcd(b, a\%b)$，其中 gcd 为最大公约数，利用递归来实现求解 gcd。

提示：定义 $gcd(a,0) = a$。

2、已知 Ackermann 函数定义如下：

$$Ack(m,n) = \begin{cases} n+1 & m = 0 \\ Ack(m-1,1) & m \neq 0, n = 0 \\ Ack(m-1, Ack(m, n-1)) & m \neq 0, n \neq 0 \end{cases}$$

（1）写出计算 Ack(m,n) 的递归算法，并根据此算法给出 Ack(2,1) 的计算过程。

（2）写出计算 Ack(m,n) 的非递归算法。

**说明：（2）这道题目可能存在一些问题，当然你直接交一个有问题的程序也没有问题，但是如果你能说明这道题目为什么不能够用非递归方式实现会更好。**

**提示：可以从 Ack(m,n) 的值域范围和所需要使用的数组内存空间考虑这个问题。**

3、完成 EOJ 上 5.1-5,2、6.1-6.2。

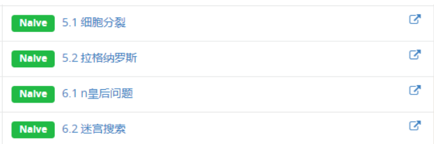


图 1: Problems that must be completed

4、EOJ 上参考完成的题目（可能会有一定难度）


图 2: Problems that are recommended

# 5 参考答案

## 5.1 选择题

B

## 5.2 简答题

1、程序用来求斐波那契数列第 $n$ 项,复杂度和斐波那契数列第 $n$ 项的值相当。由于 $a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$,所以程序的复杂度为 $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$，当然回答 $O(2^n)$ 也可以吧，但是如果写 $\Theta(2^n)$ 是不正确的。

2、复杂度为 $O(1+n^1+n^2+...+n^{m-1})$，对于 $n=1$ 的时候复杂度为 $O(m)$，对于 $n \geq 2$ 的时候复杂度可以认为是 $O(n^m)$。

## 5.3 编程题

1、按照定义写就可以了，在 C++ 中自带 __gcd 函数可以用来求两个数的最大公约数。

Listing 2: **ans1.cpp**

```
typedef long long ll;
ll gcd(ll a,ll b)
{
        return b==0?a:gcd(b,a%b);
}
```

2、按照定义写就可以了，第二题因为 Ack 函数的取值范围过大，所以直接求会造成内存溢出（用递归求解其实也会内存溢出）。

Listing 3: **ans2.cpp**

```
int Ack(int m,n)
{
        if (m==0) return(n+1);
    else if(m!=0&&n==0) return(Ack(m-1,1));
        else return(Ack(m-1,Ack(m,m-1));
}
```

Listing 4: **ans3.cpp**

```
int Ackerman(int m,int n)
{
        int akm[M][N];int i,j;
    for(j=0;j<N;j++) akm[0][j];=j+1;
    for(i=1;i<m;i++)
    {
            akm[i][0]=akm[i-1][1];
        for(j=1;j<N;j++)
            akm[i][j]=akm[i-1][akm[i][j-1]];
    }
    return(akm[m][n]);
}
```

递归调用过程请大家写完整!!!

之后作业也会有大量算法计算过程的作业，请认真完成!!!

$$
\begin{aligned}
Ack(2,1) &= Ack(1, Ack(2,0)) \\
&= Ack(1, Ack(1,1)) \\
&= Ack(1, Ack(0, Ack(1,0))) \\
&= Ack(1, Ack(0, Ack(0,1))) \\
&= Ack(1, Ack(0,2)) \\
&= Ack(1,3) \\
&= Ack(0, Ack(1,2)) \\
&= Ack(0, Ack(0, Ack(1,1))) \\
&= Ack(0, Ack(0, Ack(0, Ack(1,0)))) \\
&= Ack(0, Ack(0, Ack(0, Ack(0,1)))) \\
&= Ack(0, Ack(0, Ack(0,2))) \\
&= Ack(0, Ack(0,3)) \\
&= Ack(0,4) \\
&= 5
\end{aligned}
$$