1.2什么是算法

2019年5月3日, 星期五　　下午 01:57

1. 算法：
- 一个有限指令集
- 接受一些输入（有些情况不需要输入）
- 产生输出
- 一定在有限步骤之后终止
- 每一条指令必须
    - 有充分明确的目标，不可以尤其以
    - 计算机处理的范围之内
    - 描述不依赖语言和实现手段
2. 什么是好的算法?

a.

空间复杂度S（n）	占用存储单元的长度
时间复杂度T（n）	耗费时间的大小
n	问题规模大小

例：

输入N以内的正整数：

```
#include<stdio.h>
//输入正整数N，输出1-N的正整数
//通过递归和非递归的方法
void PrintN1(int n);
void PrintN2(int n);
int main(){
	int number;
	scanf("%d",&number);
	PrintN1(number);
	PrintN2(number);
	return 0;
}
//非递归的方法
void PrintN1(int n){
	int i;
	for(i = 1;i<=n;i++){
		printf("%d\n",i);
	}
}//占据一个变量I,一个for循环，占据的空间固定，与n无关
//S(n) =C;
//递归的方法
void PrintN2(int n){
	if(n){
		PrintN2(n-1);
		printf("%d ",n);
	}
	return;
```

}//占据c*n个空间，即空间复杂度S（n）=c*n;

//计算多项式f(x) = a0+a1X+...+an-1X(n-1)+anXn
//f1算法：从n提x开始算
//f2算法：直接a0+a1x+....开始算

```
double f1(int n,double a[],double x){
        int i;
        double p =a[n];
        for(i = n;i>0;i--){
                p = a[i-1]+x*p;
        }
        return p;
}
```

//每趟循环做一次乘法，n*1=n,一共n次，
//T(n) = C*n

```
double f2(int n,double a[],double x){
        int i ;
        double p =a[0];
        for(i = 1;i<=n;i++){
                p+=a[i]*pow(x,i);
        }
        return p;
}
```

//每趟循环做了（i-1）+1=i次乘法，一共n趟，所以1+2+3+…+n=n(n-1)/2
//$T(n) = C_1*n^2+C_2*n$

for循环的时间复杂度=循环次数 * 循环体代码的复杂度；
If-else的时间复杂度= max(循环条件判断，分支1，分支2)；