

第一题

假设 f_0, f_1, \dots, f_n 斐波那契数列, $f_0=f_1=1$, 当 $i > 1$ 时, $f_i=f(i-1) + f(i-2)$, (括号里的是下标),
1、给定最大整数 n , 设计一个 $O(\log n)$ 算法, 计算 f_n , 要求不得涉及浮点数运算, 只能进行整数运算
2、证明设计的算法时间复杂度是 $n \log n$

第二题

什么是 P 类问题, NP 问题, NP 完全问题, 分别给出他们的概念, 分别举一个例子, 同时说出它们之间的关系

第三题

第 5 页 (共 12 页)

| | |
|------|----|
| 本题分数 | 16 |
| 得分 | |

三、对于如下问题：在一个整数数组 $A[1 \dots n]$ 中，同时寻找最大值和最小值。假定 n 是 2 的整数幂。

(1) 试设计一种直接的算法求解上述问题，给出算法伪代码。

(2) 请分析此直接算法需要执行多少次数元素比较。

(3) 试运用分治递归技术对此直接算法进行改进，描述改进算法的算法思想，并给出算法伪代码。

(4) 试分析运用分治策略之后的算法需要执行的元素比较次数。

第四题

第 7 页 (共 12 页)

| | |
|------|----|
| 本题分数 | 18 |
| 得分 | |

四、将 LCS 算法应用至两个 0/1 数字串 $A=<1,0,0,1,0,1,0,1>$ 和 $B=<0,1,0,1,1,0,1,1,0>$ ，填写以下的 L 表，求解出 A 串和 B 串的最长公共子序列长度，并给出一个最长公共子序列。

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | |
| 1 | | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |

第五题

第 9 页 (共 12 页)

| | |
|------|----|
| 本题分数 | 17 |
| 得分 | |

五、问题描述: 孤岛上有 N 个人被困, 救援队派出救援船前往孤岛进行营救。每艘救援船可以承载的最大重量为 $limit$, 且每艘救援船最多可同时载两个人 (≤ 2), 为了节省救援成本, 在保证所有人都被救援的情况下, 使得派出的救援船数量最小。即给定每艘船可以承载的最大重量 $limit$, N 个人的体重: $people = [w_1, \dots, w_N] (\forall i \in \{1, \dots, N\}, 1 \leq w_i \leq limit)$, 返回可以载到 N 个人所需的最小船只数 (上述变量均为正整数)。

问题: 请写出求解该问题的主要思路和算法伪代码, 并计算该算法的复杂度, 包括时间复杂度和空间复杂度。

示例 1:
输入: $people = [1, 2], \quad limit = 3$
输出: 1
解释: 1 艘船载 (1, 2)

示例 2:
输入: $people = [3, 2, 2, 1], \quad limit = 3$
输出: 3
解释: 3 艘船分别载 (1, 2), (2) 和 (3)

示例 3:
输入: $people = [3, 5, 3, 4], \quad limit = 5$
输出: 4
解释: 4 艘船分别载 (3), (3), (4) 和 (5)

第六题

第 11 页 (共 12 页)

| | |
|------|----|
| 本题分数 | 16 |
| 得分 | |

六、问题描述:

张三手中有金币 3 枚, 抛掷硬币, 如果正面朝上就将手中金币数翻倍, 如果反面朝上就将手中金币数减 1。张三总共抛掷硬币 17 次, 其中 6 次正面朝上, 11 次反面朝上, 已知最后一次抛掷硬币的结果是反面朝上, 他手上的金币数刚好为 0。

请你计算张三抛掷硬币次序的所有可能, 硬币正面朝上记为 A, 反面朝上记为 B。则: BBABAABBBABAABBBB 就是合理的次序。请你设计算法计算所有可能方案数, 给出算法主要思路说明和伪代码。

答案：

第一题

```
1 #include<bits/stdc++.h>
2 #define ll long long
3 using namespace std;
4 const ll mod=1e9+7;
5 struct mtx
6 {
7     ll a[3][3];
8     mtx()
9     {
10         memset(a,0,sizeof(a));
11     }
12 }base,ans;
13
14 mtx operator*(mtx x,mtx b)
15 {
16     mtx c;
17     for(int i=1;i<=2;i++)
18         for(int j=1;j<=2;j++)
19             for(int k=1;k<=2;k++)
20                 {
21                     c.a[i][j]+=x.a[i][k]*b.a[k][j];
22                     c.a[i][j]%=mod;
23                 }
24     return c;
25 }
26 mtx pow(ll p)
27 {
28     mtx pp;
29     pp.a[1][1]=pp.a[2][2]=1;
30     while(p)
31     {
32         if(p&1)pp=pp*base;
33         base=base*base;
34         p>>=1;
35     }
36     return pp;
37 }
38 ll n;
39 int main()
40 {
41     //ios::sync_with_stdio(false);
42     //freopen(".txt","r",stdin);
43     base.a[1][1]=base.a[1][2]=base.a[2][1]=1;
44     cin>>n;
45     ans=pow(n);
46     cout<<ans.a[1][2];
47     return 0;
48 }
49
```

在线性代数中，类似于斐波那契数列这种递推式称为二阶递推式。我们可以用 $f(n)=af(n-1)+bf(n-2)$ 将二阶递推式一般化。只要符合这种二阶递推式的算法，都可以将算法的时间复杂度降为 $O(\log N)$ 。当然，三阶，四阶...都可以，只要得到递推公式的 n 阶矩阵即可。

其中每次矩阵运算是 $2^3=8$ 次，由于快速幂采取了分治的思想，时间复杂度为 $O(\log n)$ 。所以总时间复杂度为 $O(\log n)$ 。

第二题

p 一定是 np, npc 一定是 np, 现在还无法证明 np 是 p 类问题

①P 问题必定是 NP 问题, 但目前还无法证明 NP 问题是否是 P 问题

②NPC 问题必定是 NP 问题, 是最难的 NP 问题

③NPC 问题也必定是 NPH 问题

④NPH 问题不一定是 NP 问题

第三题

1.

```
1
2 int minx, maxx;
3 void calc(int a[], int n)
4 {
5     minx = a[1], maxx = a[1];
6     for (int i = 2; i <= n; i++)
7     {
8         if (a[i] > maxx) maxx = a[i];
9         if (a[i] < minx) minx = a[i];
10    }
11 }
```

2. 该算法总共进行了 $2 * (n-1)$ 次比较

```
3. 1 // #pragma GCC optimize(2)
2 #include <bits/stdc++.h>
3 #define ll long long
4 using namespace std;
5 template <class T>
6 void read(T& x)
7 {
8     T res = 0, f = 1; char c = getchar();
9     while (!isdigit(c)) {
10        if (c == '-') f = -1; c = getchar();
11    }
12    while (isdigit(c)) {
13        res = (res << 3) + (res << 1) + c - '0'; c = getchar();
14    }
15    x = res * f;
16 }
17 const ll N = 200000 + 10;
18
19 int a[N], minx, maxx;
20 int fmin(int a[], int l, int r)
21 {
22     if (l == r) // 只剩一个
23     {
24         return a[l];
25     }
26     int mid = (l + r) / 2;
27     return min(fmin(a, l, mid), fmin(a, mid + 1, r));
28 }
29 int fmax(int a[], int l, int r)
30 {
31     if (l == r) // 只剩一个
32     {
33         return a[l];
34     }
35     int mid = (l + r) / 2;
36     return max(fmax(a, l, mid), fmax(a, mid + 1, r));
37 }
38 int main()
39 {
40     maxx = fmax(a, 1, n);
41     minx = fmin(a, 1, n);
42     return 0;
43 }
44
```

4. 算法一共递归了 $\log n$ 层, 其中最后一层不需要比较, 所以一共比较了 $2 * (\log n - 2)$ 次

第6题

```
1. 1 // #pragma GCC optimize(2)
2 #include <bits/stdc++.h>
3 #define ll long long
4 using namespace std;
5 template <class T>
6 void read(T& x)
7 {
8     T res = 0, f = 1; char c = getchar();
9     while (!isdigit(c)) {
10         if (c == '-') f = -1; c = getchar();
11     }
12     while (isdigit(c)) {
13         res = (res << 3) + (res << 1) + c - '0'; c = getchar();
14     }
15     x = res * f;
16 }
17 const ll N = 200000 + 10;
18 int tot, a[N];
19 vector <string> ans;
20 void dfs(int step, int num, string ansp, int z, int f)
21 {
22     if (num < 0) return; // 抛弃
23     if (step == 17) // 结束递归
24     {
25         if (num == 1)
26             ans.push_back(ansp + "B"); // 最后一次为反面加一个"B"
27         return;
28     }
29     if (z > 0) // 如果正面还有
30     {
31         dfs(step + 1, num * 2, ansp + "A", z - 1, f);
32     }
33     if (f > 0) // 如果反面还有
34     {
35         dfs(step + 1, num - 1, ansp + "B", z, f - 1);
36     }
37 }
38
39 int main()
40 {
41     dfs(1, 3, "", 6, 10); // 相当于进行了16次, 6次想上, 10次向下, 结果为1
42     printf("%d\n", ans.size());
43     for (auto it : ans)
44         cout << it << endl;
45     return 0;
46 }
47 |
```

2. 答案为16种

```
1 16
2 ABBBBBABABABABB
3 BABBBABABABABABB
4 BBBBBABBBABABABABB
5 BBBBBABAABBBABABB
6 BBBBBABAABBBBABB
7 BBBBBABABABAABBBB
8 BBAABBBBABABABABB
9 BBAABBBABAABBBBABB
10 BBAABBBABAABBBBABB
11 BBAABBBABABAABBBB
12 BBABAABBBBABABABB
13 BBABAABBBABAABBBB
14 BBABAABBBABAABBBB
15 BBABAABAABBBBABB
16 BBABAABAABBBBABB
17 BBABAABAABBBBABB
```