

3.25 课堂笔记 4

课后作业：

- 1、洛谷 P2678
- 2、洛谷 P3743

上次课后作业答案：

1、

从1~n中(n<20)个整数中随机选取任意多个，输出所有可能的选择方案。

每个数可以选或者不选，所以所有可能的方案数达到了 2^n 种。

```
4 vector<int> _chosen;
5 int _n = 0;
6 void Dfs(int x) {
7     if(x == _n + 1) { // 问题的边界
8         for(int i = 0; i < _chosen.size(); i++) {
9             cout << _chosen[i] << " ";
10        }
11        cout << endl;
12        return;
13    }
14    //分支1
15    Dfs(x + 1); // 没有选择x, 继续求解子问题
16    //分支2
17    _chosen.push_back(x);
18    Dfs(x + 1); // 选择了x, 继续求解子问题
19
20    _chosen.pop_back(); // 回溯到上一个问题之前, 还原现场
21 }
22 int main () {
23     cin >> _n;
24     Dfs(1);
25 }
```

2、

从1~n中(n<20)个整数中随机选取m个，输出所有可能的选择方案。

```
4 vector<int> _chosen;
5 int _n = 0, _m = 0;
6 void Dfs(int x) {
7     // 如果已经选择超过了m个数, 或者即使把剩余数字全部选择也无法达到m个数, 就可以提前确认问题无解了
8     if(_chosen.size() > _m || _chosen.size() + (_n - x + 1) < _m) {
9         return;
10    }
11    if(x == _n + 1) { // 问题的边界
12        for(int i = 0; i < _chosen.size(); i++) {
13            cout << _chosen[i] << " ";
14        }
15        cout << endl;
16        return;
17    }
18    //分支1
19    Dfs(x + 1); // 没有选择x, 继续求解子问题
20    //分支2
21    _chosen.push_back(x);
22    Dfs(x + 1); // 选择了x, 继续求解子问题
23    _chosen.pop_back(); // 回溯到上一个问题之前, 还原现场
24 }
25 int main () {
26     cin >> _n >> _m;
27     Dfs(1);
28 }
```

3、全排列

从1~n中($n < 10$)个整数排成一行后随机打乱顺序，输出所有可能的选择方案。

```
6  int _number[10]; //按顺序依次记录被选择的整数
7  bool _use[10];   //标记被选择的整数
8  void ParseIn() {
9      cin >> _n;
10 }
11 void Dfs(int cc) {
12     if(cc == _n + 1) { //边界
13         for(int i = 1; i <= _n; i++) {
14             cout << _number[i] << " ";
15         }
16         cout << endl;
17         return;
18     }
19     for(int i = 1; i <= _n; i++) {
20         if(_use[i] == false) {
21             _number[cc] = i;
22             _use[i] = true;
23             Dfs(cc + 1);
24             _use[i] = false;
25             _number[cc] = 0;
26         }
27     }
```

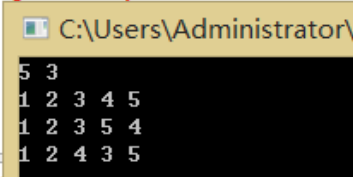
全排列

库函数：包括 `algorithm` 头文件后可调用：

`next_permutation(a+1,a+length+1);`

`prev_permutation(a+1,a+length+1);`

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int _n, _m;
5  int _list[10];
6  int main () {
7      cin >> _n >> _m;
8      for(int i = 1; i <= _n; i++) {
9          _list[i] = i;
10     }
11     for(int i = 1; i <= _n; i++) {
12         cout << _list[i] << " ";
13     }
14     cout << endl;
15
16     for(int i = 1; i <= _m - 1; i++) {
17         next_permutation(_list + 1, _list + _n + 1);
18         for(int i = 1; i <= _n; i++) {
19             cout << _list[i] << " ";
20         }
21         cout << endl;
22     }
23     return 0;
24 }
```

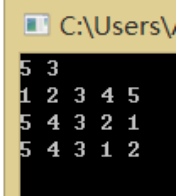


```
5 3
1 2 3 4 5
1 2 3 5 4
1 2 4 3 5
```

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  int _n, _m;
5  int _list[10];
6  int main () {
7      cin >> _n >> _m;
8      for(int i = 1; i <= _n; i++) {
9          _list[i] = i;
10     }
11     for(int i = 1; i <= _n; i++) {
12         cout << _list[i] << " ";
13     }
14     cout << endl;
15
16     for(int i = 1; i <= _m - 1; i++) {
17         prev_permutation(_list + 1, _list + _n + 1);
18         for(int i = 1; i <= _n; i++) {
19             cout << _list[i] << " ";
20         }
21         cout << endl;
22     }
23     return 0;
24 }

```



二分答案

二分查找也称折半查找 (Binary Search)，它是一种效率较高的查找方法。但是，折半查找要求线性表必须采用顺序存储结构，而且表中元素按关键字有序排列。

适用题型：

求某种条件的**最大值的**最小可能情况或者**最小值的**最大情况

解题步骤：

1.确定答案的最大值和最小值

2.判断二分所得值是否满足条件

3.可行解必须具有单调性（当 k 可行时，k+1 可行或者 k-1 可行）

注意事项：

每次都要确保 right 和 left 有一个被改变。

整数定义域上的二分模板：

```

3 int ErFen(int l, int r) {
4     int mid = 0;
5     int ans = 0;
6     while(l <= r) {
7         mid = (l + r) / 2;
8         if(Check(mid)) {
9             ans = mid;
10            l = mid + 1;
11        }
12        else {
13            r = mid - 1;
14        }
15    }
16    return ans;
17 }
18

```

实数域上的二分模板：

浮点数二分存在精度问题, 当区间端点 r 和 l 之间相差一个很小的数的时候, 认为两者相等。

```

1  #include <math.h>
2  //题意保留一位小数
3  double _eps = 1e-3;
4  //eps表示精度, 比保留小数的位数多2
5 double ErFen(double l, double r) {
6     double mid = 0;
7
8     while(fabs(l - r) > _eps) {
9         mid = (l + r) / 2.0;
10        if(Check(mid)) {
11            r = mid;
12        }
13        else {
14            l = mid;
15        }
16    }
17    return l;
18 }

```

练习 1: 洛谷 P1873

```

1  #include<iostream>
2  using namespace std;
3  const int N = 1e6 + 10;
4  int h[N];
5  int n, m, ans;
6
7  int check(int x){ //判断该高度情况下，木材总长度满不满足题目条件
8      int sum = 0;
9      for(int i = 1; i <= n; i++){
10         if(x < h[i]) sum += h[i] - x; //大于mid值，计算木材长度
11         if(sum >= m) return true;
12     }
13     return false;
14 }
15
16 int main(){
17     cin >> n >> m;
18     for(int i = 1; i <= n; i++) cin >> h[i];
19     int l = 0, r = 1e9; //确定大小值
20     while(l <= r){ //二分答案模板更新mid值
21         int mid = (l + r) / 2;
22         if(check(mid)) ans = mid, l = mid + 1;
23         else r = mid - 1;
24     }
25     cout << ans << endl;
26     return 0;
27 }
28

```

练习 2: 洛谷 P3853

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int l, r, len, ans, n, k, maxx, a[100005];
4  //通过计算可以发现 对于一个给定的距离len而言 中间分成若干个长度不大于x的段的个数为 len/x 如果len为x的倍数 则为len/x-1
5  bool check(int x){
6      int tmp=0;
7      for (int i=2;i<=n;i++){
8          tmp += (a[i]-a[i-1])/x; //需要增设的路标数量
9          if ((a[i]-a[i-1])%x==0) tmp--; //如果len为x的倍数 则为len/x-1
10     }
11     if (tmp <= k) return true;
12     return false;
13 }
14 int main(){
15     cin >> len >> n >> k;
16     for (int i=1;i<=n;i++) {
17         cin >> a[i];
18         maxx=max(maxx,a[i]-a[i-1]); //maxx是路标之间的最大距离
19     }
20     l=1;
21     r=maxx;
22     while (l<=r){
23         int mid=(l+r)/2;
24         if (check(mid)) ans = mid,r=mid-1;
25         else l=mid+1;
26     }
27     cout << ans;
28     return 0;
29 }

```

练习 3：北大 POJ3104 (网址: poj.org)

关系式推导：

$a[i] = t * k + mid - t$ (t 表示用风干机的时间, mid 是最短风干时间)

$t = (a[i] - mid) / (k - 1)$

向上取整: $t = (a[i] - mid + k - 2) / (k - 1)$

```
8 bool check(int x){
9     int cnt=0;
10    for(int i=1;i<=n;i++){
11        if(a[i]-x<=0)continue; //自然风干时间小于最短时间, 则不需要用风干机
12        cnt+=(a[i]-x+k-2)/(k-1); //计算风干机使用的总时间 (向上取整)
13        //cnt += ceil((a[i]-x)*1.0/(k-1)); //向上取整
14        if(cnt > x)return false;
15    }
16    return true;
17 }
18 int main(){
19     cin >> n;
20     int l=1,r=0,ans;
21     for(int i=1;i<=n;i++){
22         cin >> a[i];
23         r=max(r,a[i]);
24     }
25     cin >> k;
26     if(k==1){ //风干机跟自然风干效果一样, 直接输出最大值
27         cout << r;
28         return 0;
29     }
30     while(l<=r){
31         int mid=(l+r)/2;
32         if(check(mid)){
33             ans=mid; //mid表示烘干所需最短时间
34             r=mid-1; //满足条件减小最短时间
35         }
36         else{
37             l=mid+1;
38         }
39     }
40     cout << ans;
41     return 0;
42 }
```

练习 4：北大 POJ1759

由：

$$H[i] = (H[i+1] + H[i-1]) / 2 - 1$$

得到递推关系式：

$$H[i+1] = 2H[i] - H[i-1] + 2$$

再递推得到 $H[n]$ 与 $H[2]$ 之间的关系式：

$$H[n] = (n-2) * H[2] - (n-1) * H[1] + (n-1)*(n-2)$$

单调性: $H[2]$ 越小, $H[n]$ 就越小

```

4  const int INF=0x3f3f3f3f; //无穷大
5  int N;
6  double A,ans,h[1005];
7
8  bool check(double mid){
9      h[2] = mid;
10     for(int i=3; i<=N; i++){
11         h[i] = 2*h[i-1] - h[i-2] + 2; //递推关系式
12         if(h[i] < 0) return false;
13     }
14     ans = h[N]; //h[N]即最右边灯的高度。h[2]越小,h[N]也越小
15     return true;
16 }
17 int main(){
18     double l,r,mid;
19     cin >> N >> A;
20     h[1]=A;
21     l=0;
22     r=1.0*INF; //小数
23     while(r-l>1e-7){ //二分求h[2]
24         mid=(l+r)/2;
25         if(check(mid)) //mid表示h[2]的最小值
26             r=mid;
27         else
28             l=mid;
29     }
30     cout<< setiosflags(ios::fixed) << setprecision(2) << ans; //保留两位小数
31     return 0;
32 }

```