

只用暴力，稳拿省一系列。

只允许使用：

判断，循环，数组，函数，语言自带函数，素数算法，gcd，lcm 算法，一维前缀和，一维差分，简单递归。

超出范围的，或需要大量动脑的，我们都不做，只尝试输出样例

握手问题

```
1  #include <iostream>
2  using namespace std;
3  int st[51];
4  int main()
5  {
6      int ans=0;
7      for(int i=1;i<=7;i++) st[i]=1;
8      for(int i=1;i<=50;i++){
9          for(int j=i+1;j<=50;j++){
10             if(!(st[i]&&st[j])){
11                 ans++;
12             }
13         }
14     }
15     printf("%d",ans);
16     return 0;
17 }
```

```
1  public class Main {
2      public static void main(String[] args) {
3          int[] st = new int[51]; // 初始化数组
4          int ans = 0;
5
6          // 前 7 个元素设为 1
7          for (int i = 1; i <= 7; i++) {
8              st[i] = 1;
9          }
10
11          // 计算符合条件的 (i, j) 组合
12          for (int i = 1; i <= 50; i++) {
13              for (int j = i + 1; j <= 50; j++) {
14                  if (!(st[i] == 1 && st[j] == 1)) {
15                      ans++;
16                  }
17              }
18          }
19
20          System.out.println(ans);
21      }
22  }
23 }
```

```
1  def main():
```

```

2     st = [0] * 51 # 初始化数组
3     ans = 0
4
5     # 前 7 个元素设为 1
6     for i in range(1, 8):
7         st[i] = 1
8
9     # 计算符合条件的 (i, j) 组合
10    for i in range(1, 51):
11        for j in range(i + 1, 51):
12            if not (st[i] == 1 and st[j] == 1):
13                ans += 1
14
15    print(ans)
16
17 if __name__ == "__main__":
18     main()
19

```

好数

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int check(int x){
4      int cnt=1;//维护奇偶数位
5      while(x!=0){
6          int t=x%10;
7          if(cnt%2==1){//奇数
8              if(t%2==0) return 0;
9          }else{
10             if(t%2==1) return 0;
11         }
12         cnt++;
13         x/=10;
14     }
15     return 1;
16 }
17 int main(){
18     int n;
19     scanf("%d",&n);
20     int ans=0;
21     for(int i=1;i<=n;i++){
22         if(check(i)) ans++;
23     }
24     printf("%d",ans);
25 }

```

```

1  import java.util.Scanner;
2
3  public class Main {
4      // 检查数字是否符合条件
5      static boolean check(int x) {
6          int cnt = 1; // 维护奇偶数位
7          while (x != 0) {
8              int t = x % 10;
9              if (cnt % 2 == 1) { // 奇数位

```

```

10         if (t % 2 == 0) return false;
11     } else { // 偶数位
12         if (t % 2 == 1) return false;
13     }
14     cnt++;
15     x /= 10;
16 }
17 return true;
18 }
19
20 public static void main(String[] args) {
21     Scanner scanner = new Scanner(System.in);
22     int n = scanner.nextInt();
23     int ans = 0;
24
25     for (int i = 1; i <= n; i++) {
26         if (check(i)) ans++;
27     }
28
29     System.out.println(ans);
30     scanner.close();
31 }
32 }
33

```

```

1 def check(x):
2     cnt = 1 # 维护奇偶数位
3     while x != 0:
4         t = x % 10
5         if cnt % 2 == 1: # 奇数位
6             if t % 2 == 0:
7                 return False
8         else: # 偶数位
9             if t % 2 == 1:
10                 return False
11         cnt += 1
12         x //= 10
13     return True
14
15 def main():
16     n = int(input())
17     ans = sum(1 for i in range(1, n + 1) if check(i))
18     print(ans)
19
20 if __name__ == "__main__":
21     main()
22

```

R 格式

```

1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 int main() {
6     int n;

```

```

7     double d;
8     cin >> n >> d;
9
10    // 计算  $d * 2^n$ 
11    double result = d * pow(2, n);
12
13    // 四舍五入到最接近的整数
14    long long r_format = round(result);
15
16    // 输出结果
17    cout << r_format << endl;
18
19    return 0;
20 }

```

```

1  import java.math.BigDecimal;
2  import java.math.RoundingMode;
3  import java.util.Scanner;
4  //java高精度直接满分
5  public class Main {
6      public static void main(String[] args) {
7          Scanner scanner = new Scanner(System.in);
8          int n = scanner.nextInt(); // 读取 n
9          String dStr = scanner.next(); // 读取 d (作为字符串以防止精度丢失)
10         scanner.close();
11
12         // 使用 BigDecimal 进行高精度计算
13         BigDecimal d = new BigDecimal(dStr);
14         BigDecimal result = d.multiply(BigDecimal.valueOf(2).pow(n)); // d
15         * 2^n
16
17         // 四舍五入到最接近的整数
18         BigDecimal roundedResult = result.setScale(0,
19         RoundingMode.HALF_UP);
20
21         // 输出整数格式的最终结果
22         System.out.println(roundedResult.toBigInteger());
23     }
24 }

```

```

1  //python无限大也直接满分
2  n,d = input().split()
3  n = int(n)
4  r,f = d.split('.')
5  x = len(f)
6  d = int(r + f)*pow(2,n)
7  if x == 1:
8      d = str(d)
9  else:
10     d = str(d)[: -x+1]
11     if int(d[-1]) >= 5:
12         d = int(d[: -1])+1
13     else:
14         d = int(d[: -1])
15     print(d)

```

宝石组合

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <numeric>
5  using namespace std;
6
7  // 计算最小公倍数
8  long long lcm(long long a, long long b) {
9      return a / gcd(a, b) * b;
10 }
11
12 // 计算精美程度 s
13 double calcs(int a, int b, int c) {
14     long long ab = lcm(a, b);
15     long long ac = lcm(a, c);
16     long long bc = lcm(b, c);
17     long long abc = lcm(ab, c);
18     return 1.0 * a * b * c * abc / (ab * ac * bc);
19 }
20
21 // 找到最大 s 的组合
22 vector<int> findMaxS(int n, vector<int>& h) {
23     double maxS = -1;
24     vector<int> res;
25
26     for (int i = 0; i < n; ++i) {
27         for (int j = i + 1; j < n; ++j) {
28             for (int k = j + 1; k < n; ++k) {
29                 int a = h[i], b = h[j], c = h[k];
30                 double s = calcs(a, b, c);
31
32                 if (s > maxS) {
33                     maxS = s;
34                     res = {a, b, c};
35                 } else if (s == maxS) {
36                     vector<int> cur = {a, b, c};
37                     sort(cur.begin(), cur.end());
38                     sort(res.begin(), res.end());
39                     if (cur < res) {
40                         res = cur;
41                     }
42                 }
43             }
44         }
45     }
46
47     sort(res.begin(), res.end());
48     return res;
49 }
50
51 int main() {
52     int n;
53     cin >> n;
54     vector<int> h(n);
55     for (int i = 0; i < n; ++i) {
```

```

56     cin >> h[i];
57 }
58
59 vector<int> ans = findMaxS(n, h);
60 cout << ans[0] << " " << ans[1] << " " << ans[2] << endl;
61
62 return 0;
63 }

```

```

1  import java.util.*;
2
3  public class Main {
4      // 计算最大公约数
5      private static long gcd(long a, long b) {
6          while (b != 0) {
7              long temp = b;
8              b = a % b;
9              a = temp;
10         }
11         return a;
12     }
13
14     // 计算最小公倍数
15     private static long lcm(long a, long b) {
16         return a / gcd(a, b) * b;
17     }
18
19     // 计算精美程度 s
20     private static double calcs(int a, int b, int c) {
21         long ab = lcm(a, b);
22         long ac = lcm(a, c);
23         long bc = lcm(b, c);
24         long abc = lcm(ab, c);
25         return (double) (a * b * c * abc) / (ab * ac * bc);
26     }
27
28     // 找到最大 s 的组合
29     private static List<Integer> findMaxS(int n, List<Integer> h) {
30         double maxS = -1;
31         List<Integer> res = new ArrayList<>();
32
33         for (int i = 0; i < n; i++) {
34             for (int j = i + 1; j < n; j++) {
35                 for (int k = j + 1; k < n; k++) {
36                     int a = h.get(i), b = h.get(j), c = h.get(k);
37                     double s = calcs(a, b, c);
38
39                     if (s > maxS) {
40                         maxS = s;
41                         res = Arrays.asList(a, b, c);
42                     } else if (s == maxS) {
43                         List<Integer> cur = Arrays.asList(a, b, c);
44                         Collections.sort(cur);
45                         Collections.sort(res);
46                         if (cur.toString().compareTo(res.toString()) < 0) {
47                             res = cur;
48                         }
49                     }
50                 }
51             }
52         }
53         return res;
54     }
55 }

```

```

49         }
50     }
51 }
52 }
53
54     Collections.sort(res);
55     return res;
56 }
57
58     public static void main(String[] args) {
59         Scanner scanner = new Scanner(System.in);
60         int n = scanner.nextInt();
61         List<Integer> h = new ArrayList<>();
62         for (int i = 0; i < n; i++) {
63             h.add(scanner.nextInt());
64         }
65         scanner.close();
66
67         List<Integer> ans = findMaxs(n, h);
68         System.out.println(ans.get(0) + " " + ans.get(1) + " " +
ans.get(2));
69     }
70 }
71

```

```

1  import math
2  from itertools import combinations
3
4  # 计算最小公倍数
5  def lcm(a, b):
6      return a * b // math.gcd(a, b)
7
8  # 计算精美程度 s
9  def calc_s(a, b, c):
10     ab = lcm(a, b)
11     ac = lcm(a, c)
12     bc = lcm(b, c)
13     abc = lcm(ab, c)
14     return (a * b * c * abc) / (ab * ac * bc)
15
16 # 找到最大 s 的组合
17 def find_max_s(n, h):
18     max_s = -1
19     res = None
20
21     for a, b, c in combinations(h, 3):
22         s = calc_s(a, b, c)
23
24         if s > max_s:
25             max_s = s
26             res = (a, b, c)
27         elif s == max_s:
28             if res is None or sorted((a, b, c)) < sorted(res):
29                 res = (a, b, c)
30
31     return sorted(res)
32

```

```

33 # 读取输入
34 n = int(input())
35 h = list(map(int, input().split()))
36
37 # 计算结果并输出
38 ans = find_max_s(n, h)
39 print(ans[0], ans[1], ans[2])
40

```

拔河

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> PII;
5  const int N=1e5+10;
6  ll s[N];
7  int n;
8  void solve(){
9      scanf("%d",&n);
10     for(int i=1;i<=n;i++){
11         int x;
12         scanf("%d",&x);
13         s[i]=s[i-1]+x;
14     }
15     ll mins=(ll)(1e18);
16     for(int l1=1;l1<n;l1++){//[l1,r1]
17         for(int r1=l1;r1<n;r1++){
18             ll s1=s[r1]-s[l1-1];
19             for(int l2=r1+1;l2<=n;l2++){//[l2,r2]
20                 for(int r2=l2;r2<=n;r2++){
21                     ll s2=s[r2]-s[l2-1];
22                     mins=min(mins,abs(s2-s1));
23                 }
24             }
25         }
26     }
27     printf("%lld",mins);
28 }
29 int main(){
30     // ios::sync_with_stdio(false);cin.tie(0);
31     int t=1;
32     // scanf("%d",&t);
33     // cin>>t;
34     while(t--) solve();
35 }

```

```

1  import java.util.*;
2
3  public class Main {
4      static final int N = 100010;
5      static long[] s = new long[N];
6      static int n;
7
8      public static void solve(Scanner scanner) {
9          n = scanner.nextInt();

```



```

10     for (int i = 1; i <= n; i++) {
11         int x = scanner.nextInt();
12         s[i] = s[i - 1] + x;
13     }
14
15     long minDiff = Long.MAX_VALUE;
16
17     for (int l1 = 1; l1 < n; l1++) { // [l1, r1]
18         for (int r1 = l1; r1 < n; r1++) {
19             long s1 = s[r1] - s[l1 - 1];
20             for (int l2 = r1 + 1; l2 <= n; l2++) { // [l2, r2]
21                 for (int r2 = l2; r2 <= n; r2++) {
22                     long s2 = s[r2] - s[l2 - 1];
23                     minDiff = Math.min(minDiff, Math.abs(s2 - s1));
24                 }
25             }
26         }
27     }
28
29     System.out.println(minDiff);
30 }
31
32 public static void main(String[] args) {
33     Scanner scanner = new Scanner(System.in);
34     int t = 1; // 可扩展多组测试数据
35     while (t-- > 0) {
36         solve(scanner);
37     }
38     scanner.close();
39 }
40 }
41

```

```

1 def solve():
2     n = int(input())
3     s = [0] * (n + 1)
4
5     arr = list(map(int, input().split()))
6     for i in range(1, n + 1):
7         s[i] = s[i - 1] + arr[i - 1]
8
9     min_diff = float('inf')
10
11     for l1 in range(1, n): # [l1, r1]
12         for r1 in range(l1, n):
13             s1 = s[r1] - s[l1 - 1]
14             for l2 in range(r1 + 1, n + 1): # [l2, r2]
15                 for r2 in range(l2, n + 1):
16                     s2 = s[r2] - s[l2 - 1]
17                     min_diff = min(min_diff, abs(s2 - s1))
18
19     print(min_diff)
20
21 if __name__ == "__main__":
22     solve()
23

```

