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

```cpp
#include <iostream>
using namespace std;
int st[51];
int main()
{
  int ans=0;
  for(int i=1;i<=7;i++) st[i]=1;
  for(int i=1;i<=50;i++){
    for(int j=i+1;j<=50;j++){
      if(!(st[i]&&st[j])){
        ans++;
      }
    }
  }
  printf("%d",ans);
  return 0;
}
```

```java
public class Main {
    public static void main(String[] args) {
        int[] st = new int[51]; // 初始化数组
        int ans = 0;

        // 前 7 个元素设为 1
        for (int i = 1; i <= 7; i++) {
            st[i] = 1;
        }

        // 计算符合条件的 (i, j) 组合
        for (int i = 1; i <= 50; i++) {
            for (int j = i + 1; j <= 50; j++) {
                if (!(st[i] == 1 && st[j] == 1)) {
                    ans++;
                }
            }
        }

        System.out.println(ans);
    }
}

```

```python
def main():
```

```python
    st = [0] * 51  # 初始化数组
    ans = 0

    # 前 7 个元素设为 1
    for i in range(1, 8):
        st[i] = 1

    # 计算符合条件的 (i, j) 组合
    for i in range(1, 51):
        for j in range(i + 1, 51):
            if not (st[i] == 1 and st[j] == 1):
                ans += 1

    print(ans)

if __name__ == "__main__":
    main()

```

好数

```cpp
#include<bits/stdc++.h>
using namespace std;
int check(int x){
  int cnt=1;//维护奇偶数位
  while(x!=0){
    int t=x%10;
    if(cnt%2==1){//奇数
      if(t%2==0) return 0;
    }else{
      if(t%2==1) return 0;
    }
    cnt++;
    x/=10;
  }
  return 1;
}
int main(){
  int n;
  scanf("%d",&n);
  int ans=0;
  for(int i=1;i<=n;i++){
    if(check(i)) ans++;
  }
  printf("%d",ans);
}
```

```java
import java.util.Scanner;

public class Main {
    // 检查数字是否符合条件
    static boolean check(int x) {
        int cnt = 1; // 维护奇偶数位
        while (x != 0) {
            int t = x % 10;
            if (cnt % 2 == 1) { // 奇数位
```

```
                if (t % 2 == 0) return false;
            } else { // 偶数位
                if (t % 2 == 1) return false;
            }
            cnt++;
            x /= 10;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int ans = 0;

        for (int i = 1; i <= n; i++) {
            if (check(i)) ans++;
        }

        System.out.println(ans);
        scanner.close();
    }
}

```

```
def check(x):
    cnt = 1  # 维护奇偶数位
    while x != 0:
        t = x % 10
        if cnt % 2 == 1:  # 奇数位
            if t % 2 == 0:
                return False
        else:  # 偶数位
            if t % 2 == 1:
                return False
        cnt += 1
        x //= 10
    return True

def main():
    n = int(input())
    ans = sum(1 for i in range(1, n + 1) if check(i))
    print(ans)

if __name__ == "__main__":
    main()

```

R 格式

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    int n;
```

```
    double d;
    cin >> n >> d;

    // 计算 d * 2^n
    double result = d * pow(2, n);

    // 四舍五入到最接近的整数
    long long r_format = round(result);

    // 输出结果
    cout << r_format << endl;

    return 0;
}
```

```
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;
//java高精度直接满分
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();  // 读取 n
        String dStr = scanner.next();  // 读取 d（作为字符串以防止精度丢失）
        scanner.close();

        // 使用 BigDecimal 进行高精度计算
        BigDecimal d = new BigDecimal(dStr);
        BigDecimal result = d.multiply(BigDecimal.valueOf(2).pow(n)); // d
* 2^n

        // 四舍五入到最接近的整数
        BigDecimal roundedResult = result.setScale(0,
RoundingMode.HALF_UP);

        // 输出整数格式的最终结果
        System.out.println(roundedResult.toBigInteger());
    }
}

```

```
//python无限大也直接满分
n,d = input().split()
n = int(n)
r,f = d.split('.')
x = len(f)
d = int(r + f)*pow(2,n)
if x == 1:
    d = str(d)
else:
    d = str(d)[:-x+1]
if int(d[-1]) >= 5:
    d = int(d[:-1])+1
else:
    d = int(d[:-1])
print(d)
```

宝石组合

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
using namespace std;

// 计算最小公倍数
long long lcm(long long a, long long b) {
    return a / gcd(a, b) * b;
}

// 计算精美程度 S
double calcS(int a, int b, int c) {
    long long ab = lcm(a, b);
    long long ac = lcm(a, c);
    long long bc = lcm(b, c);
    long long abc = lcm(ab, c);
    return 1.0 * a * b * c * abc / (ab * ac * bc);
}

// 找到最大 S 的组合
vector<int> findMaxS(int n, vector<int>& h) {
    double maxS = -1;
    vector<int> res;

    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            for (int k = j + 1; k < n; ++k) {
                int a = h[i], b = h[j], c = h[k];
                double s = calcS(a, b, c);

                if (s > maxS) {
                    maxS = s;
                    res = {a, b, c};
                } else if (s == maxS) {
                    vector<int> cur = {a, b, c};
                    sort(cur.begin(), cur.end());
                    sort(res.begin(), res.end());
                    if (cur < res) {
                        res = cur;
                    }
                }
            }
        }
    }

    sort(res.begin(), res.end());
    return res;
}

int main() {
    int n;
    cin >> n;
    vector<int> h(n);
    for (int i = 0; i < n; ++i) {
```

```
        cin >> h[i];
    }

    vector<int> ans = findMaxS(n, h);
    cout << ans[0] << " " << ans[1] << " " << ans[2] << endl;

    return 0;
}
```

```
import java.util.*;

public class Main {
    // 计算最大公约数
    private static long gcd(long a, long b) {
        while (b != 0) {
            long temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    // 计算最小公倍数
    private static long lcm(long a, long b) {
        return a / gcd(a, b) * b;
    }

    // 计算精美程度 S
    private static double calcS(int a, int b, int c) {
        long ab = lcm(a, b);
        long ac = lcm(a, c);
        long bc = lcm(b, c);
        long abc = lcm(ab, c);
        return (double) (a * b * c * abc) / (ab * ac * bc);
    }

    // 找到最大 S 的组合
    private static List<Integer> findMaxS(int n, List<Integer> h) {
        double maxS = -1;
        List<Integer> res = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                for (int k = j + 1; k < n; k++) {
                    int a = h.get(i), b = h.get(j), c = h.get(k);
                    double s = calcS(a, b, c);

                    if (s > maxS) {
                        maxS = s;
                        res = Arrays.asList(a, b, c);
                    } else if (s == maxS) {
                        List<Integer> cur = Arrays.asList(a, b, c);
                        Collections.sort(cur);
                        Collections.sort(res);
                        if (cur.toString().compareTo(res.toString()) < 0) {
                            res = cur;
                        }
```

```
                    }
                }
            }
        }

        Collections.sort(res);
        return res;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        List<Integer> h = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            h.add(scanner.nextInt());
        }
        scanner.close();

        List<Integer> ans = findMaxS(n, h);
        System.out.println(ans.get(0) + " " + ans.get(1) + " " +
ans.get(2));
    }
}

```

```
import math
from itertools import combinations

# 计算最小公倍数
def lcm(a, b):
    return a * b // math.gcd(a, b)

# 计算精美程度 S
def calc_s(a, b, c):
    ab = lcm(a, b)
    ac = lcm(a, c)
    bc = lcm(b, c)
    abc = lcm(ab, c)
    return (a * b * c * abc) / (ab * ac * bc)

# 找到最大 S 的组合
def find_max_s(n, h):
    max_s = -1
    res = None

    for a, b, c in combinations(h, 3):
        s = calc_s(a, b, c)

        if s > max_s:
            max_s = s
            res = (a, b, c)
        elif s == max_s:
            if res is None or sorted((a, b, c)) < sorted(res):
                res = (a, b, c)

    return sorted(res)

```

```python
# 读取输入
n = int(input())
h = list(map(int, input().split()))

# 计算结果并输出
ans = find_max_s(n, h)
print(ans[0], ans[1], ans[2])

```

拔河

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
const int N=1e5+10;
ll s[N];
int n;
void solve(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        int x;
        scanf("%d",&x);
        s[i]=s[i-1]+x;
    }
    ll mins=(ll)(1e18);
    for(int l1=1;l1<n;l1++){//[l1,r1]
        for(int r1=l1;r1<n;r1++){
            ll s1=s[r1]-s[l1-1];
            for(int l2=r1+1;l2<=n;l2++){//[l2,r2]
                for(int r2=l2;r2<=n;r2++){
                    ll s2=s[r2]-s[l2-1];
                    mins=min(mins,abs(s2-s1));
                }
            }
        }
    }
    printf("%lld",mins);
}
int main(){
    // ios::sync_with_stdio(false);cin.tie(0);
    int t=1;
    // scanf("%d",&t);
    // cin>>t;
    while(t--) solve();
}
```

```java
import java.util.*;

public class Main {
    static final int N = 100010;
    static long[] s = new long[N];
    static int n;

    public static void solve(Scanner scanner) {
        n = scanner.nextInt();
```

```java
        for (int i = 1; i <= n; i++) {
            int x = scanner.nextInt();
            s[i] = s[i - 1] + x;
        }

        long minDiff = Long.MAX_VALUE;

        for (int l1 = 1; l1 < n; l1++) { // [l1, r1]
            for (int r1 = l1; r1 < n; r1++) {
                long s1 = s[r1] - s[l1 - 1];
                for (int l2 = r1 + 1; l2 <= n; l2++) { // [l2, r2]
                    for (int r2 = l2; r2 <= n; r2++) {
                        long s2 = s[r2] - s[l2 - 1];
                        minDiff = Math.min(minDiff, Math.abs(s2 - s1));
                    }
                }
            }
        }

        System.out.println(minDiff);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int t = 1; // 可扩展多组测试数据
        while (t-- > 0) {
            solve(scanner);
        }
        scanner.close();
    }
}

```

```python
def solve():
    n = int(input())
    s = [0] * (n + 1)

    arr = list(map(int, input().split()))
    for i in range(1, n + 1):
        s[i] = s[i - 1] + arr[i - 1]

    min_diff = float('inf')

    for l1 in range(1, n):  # [l1, r1]
        for r1 in range(l1, n):
            s1 = s[r1] - s[l1 - 1]
            for l2 in range(r1 + 1, n + 1):  # [l2, r2]
                for r2 in range(l2, n + 1):
                    s2 = s[r2] - s[l2 - 1]
                    min_diff = min(min_diff, abs(s2 - s1))

    print(min_diff)

if __name__ == "__main__":
    solve()

```