



荃灣官立中學 聖馬可中學

2026/27 SDIC 初赛练习

限时 2 小时

总览

部分	类型	占分
一	单项选择题	30
二	阅读程序	40
三	完善程序	30
总分		100

考生注意事项

- 试题共有 18 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算机、手机、电子词典、电子手表等）或查阅任何书籍资料。

— 单项选择题

占分

30

- 共 15 题，每题 2 分，共计 30 分
- 每题有且仅有一个正确选项

1 `g++ {var} brute brute.cpp`

上述指令用于将 C++ 档案 `brute.cpp` 编译为名为 `brute` 的执行档。在该指令中，`{var}` 应该被替换为哪个选项？

- A `-o`
- B `-std`
- C `-O2`
- D `-nm`

2 考虑对 N 个数字进行排序，下列哪种排序演算法在最坏情况下具有最佳的渐近时间复杂度？

- A 快速排序
- B 归并排序
- C 插入排序
- D 冒泡排序

3 下列哪一组操作能完成从双向循环链表中删除节点 `x` 的效果？（其中，`next` 域为节点的直接后继，`prev` 域为节点的直接前驱）

- A `x->prev->next = x->next; x->next->prev = x->prev;`
- B `x->prev->prev = x->next; x->next->next = x->prev;`
- C `x->next->prev = x->next; x->prev->next = x->prev;`
- D `x->prev->prev = x->prev; x->next->next = x->next;`

4 将元素 $\{7, 3, 6, 8, 2, 1, 4, 5\}$ 依序插入一棵二元搜寻树。若规定根节点的深度为 1，则所有节点的深度总和是多少？

- A 21
- B 22
- C 23
- D 24

- 5 在 KMP 演算法中，对于模式串 P ，其 next 数组 $\text{next}[i]$ 定义为模式串 $P[1 \dots i]$ 最长公共前后缀的长度，且阵列下标从 1 开始。若 P 长度为 6，且 $P[1 \dots 3] = \text{abb}$ ，则有多少种不同 next 数组？
- A 4
B 5
C 12
D 13
- 6 某演算法的时间复杂度的递归式为 $T(n) = 2T(\sqrt{n}) + O(\log n)$ ，则其渐进时间复杂度为？
- A $O(\log(n))$
B $O(\log(n) \log(\log(n)))$
C $O(\log^2(n))$
D $O(n)$
- 7 定义 $G(A)$ 为一个带权完全图，其中任意一对顶点 (i, j) 之间的边权为 $A_i + A_j + |i - j|$ 。已知 P 是 1 到 7 的排列，对于所有可能的 $G(P)$ ，其最小生成树 (MST) 边权和的最大可能值与最小可能值之差为？
- A 0
B 6
C 9
D 10
- 8 将 9 块相同的糖果分给 4 个小孩（允许有小孩分不到糖果），要求分到糖果最多的小孩比分到最少的小孩至少多 3 块，有多少种不同的分配方案？
- A 210
B 216
C 256
D 292
- 9 抽屉里有 3 只红袜子、4 只白袜子和 2 只黑袜子。最少需要抽出多少只袜子，才能保证凑出两双同色的袜子（注：两双袜子可以是同一种颜色，也可以是不同颜色）？
- A 4
B 5
C 6
D 7

- 10 對於初始全為 0、大小為 10^{12} 的數組，若要支援強制在線的區間加與單點查詢操作，下列哪種資料結構/方法最為適合？
- A 動態開點線段樹
 - B 離散化
 - C 分塊
 - D 差分數組
- 11 由 10 個長度為 5 的 01 字串構成的 Trie 樹（包含根節點），其最大節點數為？
- A 46
 - B 47
 - C 50
 - D 1024
- 12 關於笛卡爾樹，下列哪一項說法是錯誤的？
- A 使用單調棧能實現笛卡爾樹的構建，其時間複雜度為 $O(n)$ 。
 - B 笛卡爾樹滿足堆的性質。
 - C 笛卡爾樹是一棵二元樹。
 - D 由長度為 n 的隨機排列所建構的笛卡爾樹，其期望樹深為 $O(\sqrt{n})$ 。
- 13 具有 6 個頂点的非同構、包含歐拉回路的二分簡單連通圖共有幾個？
- A 1
 - B 2
 - C 3
 - D 4
- 14 透過記憶化優化遞推公式後，下列哪一個公式的時間複雜度提升最為顯著？（公式適用於正整數 $n > 1$ ，已知 $F(0) = F(1) = 1$ ）
- A $F(n) = F(\lfloor \frac{n}{2} \rfloor) + F(\lfloor \frac{n}{3} \rfloor)$
 - B $F(n) = F(n-1) + F(n-2)$
 - C $F(n) = F(n-1) * n$
 - D $F(n) = \sum F(2^i)$ 當 $n-1$ 的二進位第 i 位為 1 時



- 15 定义 $G(L, R)$ 为一个无向图，其顶点编号为闭区间 $[L, R]$ 内的所有整数，且当且仅当两个整数互质时，两点之间存在一条无向边。求满足 $1 \leq L < R \leq 100$ 且至少包含一条割边的 $G(L, R)$ 的数量。
- A 49
 - B 125
 - C 148
 - D 224

二 阅读程序

占分

40

- 程序输入不超过数组或字符串定义的范围
- 判断题正确填√，错误填×
- 除特殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分

(1)

C++

```
1  #include <algorithm>
2  #include <cstdio>
3  using namespace std;
4
5  const int N = 1000;
6  int arr[N], brr[N], sw = 0;
7
8  void selfsort(int l, int r, int id) {
9      if (arr[l] > arr[r]) {
10         swap(arr[l], arr[r]);
11         sw++;
12     }
13     if (id < 0 || r - l + 1 < brr[id]) return;
14     selfsort(l, l + brr[id] - 1, id - 1);
15     selfsort(r - brr[id] + 1, r, id - 1);
16     selfsort(l, l + brr[id] - 1, id - 1);
17 }
18
19 int main() {
20     int n, m;
21     scanf("%d", &n);
22     for (int i = 0; i < n; i++) scanf("%d", arr + i);
23     scanf("%d", &m);
24     for (int i = 0; i < m; i++) scanf("%d", brr + i);
25     sort(brr, brr + m);
26     selfsort(0, n - 1, m - 1);
27     printf("%d\n", sw);
28 }
```

- 16 当 brr 包含 1，可能会出现运行时错误。
- 17 当 $m \geq \lceil \log_2(n) \rceil$ 时，程序终止时，保证 arr 是有序的。
- 18 对于所有使程序保证对数组 arr 排序的 brr ，算法的最坏时间复杂度永远不会优于冒泡排序。



19 当输入为 5 4 2 5 1 3 2 2 3 时，输出结果为 _____。

- A 4
- B 5
- C 6
- D 7

20 (2.5 分) 删除第 25 行的 `sort(brr, brr + m);`，则输出会 _____。

- A 变大或不变
- B 变小或不变
- C 一定不变
- D 不确定

21 当输入的 $n = 6$ 且 $brr = \{2, 4\}$ 时，输出的最大值为 _____。

- A 3
- B 4
- C 5
- D 6

(2)

C++

```
1 #include <cstdio>
2 #include <vector>
3 #include <queue>
4 using namespace std;
5
6 struct DS {
7     priority_queue<int> pq1;
8     priority_queue<int, vector<int>, greater<int>> pq2;
9     void insert(int x) {
10         if (pq1.empty() || x <= pq1.top()) pq1.push(x);
11         else pq2.push(x);
12         while (pq1.size() > pq2.size() + 1) pq2.push(pq1.top()), pq1.pop();
13         while (pq2.size() > pq1.size()) pq1.push(pq2.top()), pq2.pop();
14     }
15     int query() {
16         return pq1.top();
17     }
18 };
19
20 int dp[105][105], arr[105];
21 int main() {
22     int n, k;
23     scanf("%d %d", &n, &k);
24     for (int i = 1; i <= n; i++) scanf("%d", arr + i);
25     for (int i = 1; i <= n; i++) {
26         fill(dp[i], dp[i] + 105, -1e9);
27         DS st;
28         for (int j = i; j >= 1; j--) {
29             st.insert(arr[j]);
30             int val = st.query();
31             for (int l = 1; l <= k; l++) {
32                 dp[i][l] = max(dp[i][l], dp[j - 1][l - 1] + val);
33             }
34         }
35     }
36     printf("%d\n", dp[n][k]);
37 }
```

22 将第 12 行的 `pq1.size() > pq2.size() + 1` 改为 `pq1.size() > pq2.size()` 后，代码输出的结果不会受到影响。

23 当输入的 $k = 1$ 时，`arr` 中至少有 $\lfloor \frac{n}{2} \rfloor$ 个元素大于输出。

24 当输入为 $k = n$ 时，输出为 `arr` 中元素的总和。



25 当输入为 8 2 3 1 4 1 5 9 2 6 时，程式输出的答案为 _____ 。

- A 7
- B 9
- C 10
- D 15

26 当输入为 8 3 -3 -1 -4 -1 -5 -9 -2 -6 时，程式输出的答案是 _____ 。

- A -1
- B -4
- C -6
- D -10

27 假设 k 是 $O(n)$ ，代码的时间复杂度是 _____ 。

- A $O(n^2)$
- B $O(n^2 \log(n))$
- C $O(n^3)$
- D $O(n^3 \log(n))$

(3)

C++

```
1  #include <cmath>
2  #include <cstdio>
3  #include <numeric>
4  #include <vector>
5  int cnt = 0;
6  int n, k;
7  int check(std::vector<int> L, std::vector<int> R) {
8      cnt++;
9      if (L.size() > R.size()) return -1;
10     if (L.size() < R.size()) return 1;
11     for (int i = 0; i < L.size(); i++) {
12         if(L[i] == abs(k)) return -k / abs(k);
13         if(R[i] == abs(k)) return k / abs(k);
14     }
15     return 0;
16 }
17 bool assert_ans(int ans) {
18     if (ans == k) {
19         printf("You are Right using %d checks\n", cnt);
20         return true;
21     }
22     printf("Wrong answer!\n");
23     return false;
24 }
25 void guess1(int n) {
26     std::vector<int> A(n);
27     iota(A.begin(), A.end(), 1);
28     if (check({A[0]}, {A[1]}) != 0 && check({A[0]}, {A[2]}) != 0) {
29         std::swap(A[0], A[1]);
30     }
31     for (int i = 1; i < A.size(); i++) {
32         int tmp = check({A[0]}, {A[i]});
33         if (tmp != 0) {
34             assert_ans(A[i] * tmp);
35             return;
36         }
37     }
38     assert_ans(0);
39     return;
40 }
41 void guess2(int n) {
42     std::vector<int> A(n);
43     iota(A.begin(), A.end(), 1);
44     if (check({A[0]}, {A.back()}) != 0 && check({A[1]}, {A.back()}) != 0) {
45         std::swap(A[0], A.back());
46     }
47     int b = A.back();
48     A.pop_back();
```

```
49     while (A.size() >= 3) {
50         std::vector<int> G[3], left;
51         while (A.size() % 3) {
52             left.push_back(A.back());
53             A.pop_back();
54         }
55         for (int i = 0; i < A.size(); i++) {
56             G[i % 3].push_back(A[i]);
57         }
58         A = G[2];
59         if (check(G[0], G[1])) {
60             if (check(G[0], G[2])) A = G[0];
61             else A = G[1];
62         }
63         while (left.size()) {
64             A.push_back(left.back());
65             left.pop_back();
66         }
67     }
68     assert_ans(A[0] * check({b}, {A[0]}));
69     return;
70 }
71 int main() {
72     scanf("%d %d", &n, &k);
73     int t;
74     scanf("%d", &t);
75     if (t == 1) guess1(n);
76     else guess2(n);
77     return 0;
78 }
```

(注意：下述的「猜测数」为调用 `check` 函数的次数 (即 `cnt` 的值)；「猜测正确」的含义为 `assert_ans` 函数 `return true` (执行第 20 行所在分支) 的情况；所有输入保证 $-n \leq k \leq n$ 且 $3 \leq n \leq 100$ 。)

28 当输入为 6 5 1 时，猜测次数为 5；输入为 6 5 2 时，猜测次数为 4。

29 不管输入的 n 和 k 具体为多少， $t = 2$ 时的猜测数总是小于等于 $t = 1$ 时的猜测数。

30 不管 $t = 1$ 或 $t = 2$ ，程式都一定会猜到正确结果。



- 31 函数 `guess1` 在终止时，`cnt` 的值最少为 _____ 。
- A 1
 - B 2
 - C 3
 - D 4
- 32 函数 `guess2` 在运行过程中，最多使用的猜测次数的量级为 _____ 。
- A $O(n)$
 - B $O(n^2)$
 - C $O(\sqrt{n})$
 - D $O(\log n)$
- 33 当输入的 $n = 100$ 的时候，代码中 $t = 1$ 和 $t = 2$ 分别需要的猜测次数最多分别为 _____ 。
- A 100, 14
 - B 100, 13
 - C 99, 14
 - D 99, 13

三 完善程序

占分

30

- 单选题，每小题 3 分，共计 30 分

(1) (**k -有界连通对**) 给定一个含 n 个点、 m 条边的带权无向连通图。对于一条 u 到 v 的路径，该路径被称为「强 k -有界」当且仅当路径上所有边的权重至多为 k ，该路径被称为「弱 k -有界」当且仅当路径上至多有一条边的权重大于 k 。注意，「强 k -有界」路径总是「弱 k -有界」的。对于点对 (u, v) ，该对被称为「强 k -有界连通」当且仅当存在至少一条由 u 到 v 的「强 k -有界」路径；同样地，该对被称为「弱 k -有界连通」当且仅当存在至少一条由 u 到 v 的「弱 k -有界」路径。注意，点对 (u, u) 是「强 k -有界连通」且「弱 k -有界连通」的。给定 Q 个查询，对于不同的 k ，分别求出图中「强 k -有界连通」的有序点对的数量和「弱 k -有界连通」的有序点对的数量。上述参数满足 $1 \leq n, m \leq 1000$, $1 \leq q \leq 10$, $1 \leq u, v \leq n$ 和 $1 \leq w, k \leq 10^5$ 。

以下代码求解了上述问题。试补全程序。

C++

```
1 #include <algorithm>
2 #include <array>
3 #include <cstdio>
4 #include <deque>
5 #include <numeric>
6 #include <vector>
7 int p[2005], W[2005], cnt[100005];
8 bool vis[1005];
9 std::vector<std::pair<int, int>> G[1005];
10 std::vector<int> T[2005];
11 int find(int u) {
12     return p[u] == u ? _____ 1 _____;
13 }
14 int dfs(int u) {
15     if (T[u].empty()) return 1;
16     int a = dfs(T[u][0]), b = dfs(T[u][1]);
17     cnt[W[u]] += _____ 2 _____;
18     return a + b;
19 }
20 int main() {
21     int n, m, q;
22     scanf("%d %d %d", &n, &m, &q);
23     std::iota(p, p + 2005, 0);
24     std::vector<std::array<int, 3>> eds;
25     for (int i = 0; i < m; i++) {
26         int u, v, w;
27         scanf("%d %d %d", &u, &v, &w);
28         G[u].emplace_back(v, w);
29         G[v].emplace_back(u, w);
30         eds.push_back({w, u, v});
31     }
32     sort(eds.begin(), eds.end());
```

```
33 int ttl = n;
34 for (auto [w, u, v] : eds) {
35     u = find(u); v = find(v);
36     if (u == v) continue;
37     T[++ttl] = {u, v};
38     p[u] = p[v] = ttl;
39     W[ttl] = w;
40 }
41 dfs(ttl);
42 for (int i = 1; i <= 100000; i++) cnt[i] = 3;
43 while (q--) {
44     int k;
45     scanf("%d", &k);
46     int ans = 0;
47     for (int i = 1; i <= n; i++) {
48         std::fill(vis, vis + n + 1, 0);
49         std::deque<std::pair<int, int>> dq;
50         dq.emplace_back(i, 0);
51         while (!dq.empty()) {
52             auto [u, d] = dq.front(); dq.pop_front();
53             if (4) continue;
54             vis[u] = true;
55             ans++;
56             for (auto [v, w] : G[u]) {
57                 if (w <= k) dq.emplace_front(v, d);
58                 else dq.emplace_back(5);
59             }
60         }
61     }
62     printf("%d %d\n", cnt[k] * 2 + n, ans);
63 }
64 }
```

34 1 处应填

- A u : p[u] = find(u)
- B u : p[u] = find(p[u])
- C p[u] = find(u) : u
- D p[u] = find(p[u]) : u

35 2 处应填

- A (a - 1) * (b - 1)
- B a * b - a - b
- C a * b - 1
- D a * b

36 _____ 3 _____ 处应填

- A `cnt[i - 1] + cnt[i]`
- B `cnt[i - 1] * cnt[i]`
- C `cnt[i] * cnt[i]`
- D `cnt[i] * (cnt[i] + 1) / 2`

37 _____ 4 _____ 处应填

- A `vis[u] || d > 1`
- B `vis[u] || d > k`
- C `vis[u] && d > 1`
- D `vis[u] && d > k`

38 _____ 5 _____ 处应填

- A `v, w`
- B `v, d + 1`
- C `v, d + w`
- D `v, w / k`

(2) (最小互质代价) 给定一个长度为 n 的数组 arr 。定义数组 A ，其长度为 $n \times m$ ，且 $A_i = arr_{i \bmod n}$ (下标从 0 开始)。设 B 为长度为 $n \times m$ 的数组，满足 $B_i \geq 0$ 和 $|A_i - B_i| \leq 40$ ，且对所有 $i > 0$ ， B_{i-1} 与 B_i 互质 (即 $\gcd(B_{i-1}, B_i) = 1$)。若代价定义为 $\sum |A_i - B_i|$ ，求最小代价。上述参数满足 $1 \leq n \leq 100$ 和 $1 \leq m, arr_i \leq 10^9$ 。保证至少存在一个合法的 B 。

以下代码求解了上述问题。试补全程序。

C++

```
1 #include <algorithm>
2 #include <cstdio>
3 #include <numeric>
4 #include <vector>
5 const int RNG = 40, R = 2 * RNG + 1;
6 const long long INF = 1e18;
7 int n, m;
8 int arr[105];
9 inline bool check(int idx, int j, int k) {
10     int v1 = arr[idx] + j - RNG;
11     int v2 = arr[idx + 1] + k - RNG;
12     return (v1 >= 0 && v2 >= 0 && std::gcd(v1, v2) == 1);
13 }
14 struct Matrix {
15     long long mat[R][R];
16     Matrix() {
17         for (int i = 0; i < R; ++i)
18             for (int j = 0; j < R; ++j)
19                 mat[i][j] = INF;
20     }
21     static Matrix mul(const Matrix& A, const Matrix& B) {
22         Matrix C;
23         for (int k = 0; k < R; ++k)
24             for (int i = 0; i < R; ++i)
25                 for (int j = 0; j < R; ++j)
26                     C.mat[i][j] = _____;
27         return C;
28     }
29 };
30 Matrix qpow(Matrix A, long long p) {
31     Matrix res = A;
32     p--;
33     while (p > 0) {
34         if (p & 1) res = Matrix::mul(res, A);
35         A = _____;
36         p >>= 1;
37     }
38     return res;
39 }
40 std::vector<long long> run_dp(int begin_val) {
41     std::vector<long long> dp(R, INF);
42     for (int k = 0; k < R; ++k) {
43         if (begin_val == -1 || check(n - 1, begin_val, k))
```

```
44     dp[k] = abs(k - RNG);
45 }
46 for (int j = 1; j < n; ++j) {
47     std::vector<long long> ndp(R, INF);
48     for (int k = 0; k < R; ++k)
49         for (int l = 0; l < R; ++l)
50             if (____ 3 ____ )
51                 ndp[k] = std::min(ndp[k], ____ 4 ____ );
52     dp = move(ndp);
53 }
54 return dp;
55 }
56 int main() {
57     scanf("%d %d", &n, &m);
58     for (int i = 0; i < n; ++i) scanf("%d", arr + i);
59     arr[n] = arr[0];
60     std::vector<long long> vec = run_dp(-1);
61     if (m == 1) {
62         printf("%lld\n", *min_element(vec.begin(), vec.end()));
63         return 0;
64     }
65     Matrix T;
66     for (int i = 0; i < R; ++i) {
67         if (arr[n - 1] + i - RNG < 0) continue;
68         std::vector<long long> dp_i = run_dp(i);
69         for (int j = 0; j < R; ++j) T.mat[j][i] = dp_i[j];
70     }
71     Matrix res = ____ 5 ____ ;
72     long long ans = INF;
73     for (int i = 0; i < R; ++i)
74         for (int j = 0; j < R; ++j)
75             ans = std::min(ans, vec[i] + res.mat[j][i]);
76     printf("%lld\n", ans);
77     return 0;
78 }
```

39 _____ 1 _____ 处应填

- A C.mat[i][j] + std::min(A.mat[i][k], B.mat[k][j])
- B C.mat[i][j] + A.mat[i][k] * B.mat[k][j]
- C std::min(C.mat[i][j], A.mat[i][k] + B.mat[k][j])
- D std::min(C.mat[i][j], A.mat[i][k] * B.mat[k][j])

40 _____ 2 _____ 处应填

- A `Matrix::mul(A, 2)`
- B `Matrix::mul(A, A)`
- C `Matrix::mul(res, 2)`
- D `Matrix::mul(res, res)`

41 _____ 3 _____ 处应填

- A `check(j - 1, k, 1)`
- B `check(j - 1, 1, k)`
- C `check(j, k, 1)`
- D `check(j, 1, k)`

42 _____ 4 _____ 处应填

- A `dp[k] + abs(k - RNG)`
- B `dp[k] + abs(1 - RNG)`
- C `dp[1] + abs(k - RNG)`
- D `dp[1] + abs(1 - RNG)`

43 _____ 5 _____ 处应填

- A `qpow(T, m - 1)`
- B `qpow(T, m)`
- C `qpow(T, (long long)n * m - 1)`
- D `qpow(T, (long long)n * m)`