DỰ ÁN CHICKEN MINDS
BAN CHUYÊN MÔN TIN HỌC

KỲ THI THỬ TUYỂN SINH LỚP 10 THPT CHUYÊN
NĂM HỌC 2024 – 2025

HƯỚNG DẪN CHẤM THI
Đề thi thử đợt 1

Môn thi: **TIN HỌC**
Ngày thi: **16/3/2024 – 30/3/2024**
Thời gian làm bài: **150 phút** (không kể thời gian phát đề)
Hướng dẫn chấm thi gồm 06 trang

Tổng quan đề thi

	Tên bài	Tập tin dữ liệu	Tập tin kết quả	Hạn chế thời gian	Hạn chế bộ nhớ
Bài 1	Không đẹp	KHONGDEP.INP	KHONGDEP.OUT	2 giây	512MB
Bài 2	Mắt xích	MATXICH.INP	MATXICH.OUT	2 giây	512MB
Bài 3	Diện tích	AREA.INP	AREA.OUT	2 giây	512MB

I. Hướng dẫn chung

1. Bài thi của thí sinh được chấm thi bằng phần mềm chấm thi trực tuyến (online judge) trên website `https://oj.giftedbat.edu.vn/`, sử dụng bộ test của Tổ chức The Gifted Battlefield – Ban Tin học, đúng với đáp án và biểu điểm được nêu trong hướng dẫn chấm thi.
2. Điểm bài thi được xuất từ phần mềm chấm thi; không quy tròn điểm thành phần của từng câu và điểm của bài thi.

II. Lời giải

Bài 1. Không đẹp – KHONGDEP (3,5 điểm)

Subtask 1

Giới hạn: $n \leq 1000$.
Với giới hạn này, ta có thể sử dụng hai vòng for để đếm tất cả cặp i, j thoả điều kiện $i < j$ và $S[i] > S[j]$.
Độ phức tạp: $O(n^2)$.

Subtask 2

Giới hạn: xâu S chỉ gồm các kí tự `a` và `b`.

- Với subtask này, ta sẽ xét qua từng vị trí i trong xâu và đếm xem có bao nhiêu vị trí j $(0 \leq j < i)$ sao cho $S[j] > S[i]$:
  - Nếu $S[i] =$ 'a', ta cần biết có bao nhiêu vị trí j sao cho với $0 \leq j < i$ và $S[j] =$ 'b'.
  - Nếu $S[i] =$ 'b', ta bỏ qua và không cần quan tâm.
- Như vậy, trong quá trình duyệt i từ 0 đến $n-1$, ta cần sử dụng một biến đếm cnt lưu số ký tự 'b' đã xuất hiện đến vị trí i hiện tại. Phần cài đặt sẽ có dạng:

```
long long res = 0, cnt = 0;
for (int i = 0; i < n; i++) {
    if (s[i] == 'a') res += cnt;
    else cnt++;
}
```

Độ phức tạp: $O(n)$.

Subtask 3

Yêu cầu kiến thức: Mảng đếm, mảng cộng dồn.
Ở subtask cuối, ta sẽ sử dụng ý tưởng tương tự như **Subtask 2**. Với ký tự $S[i]$ tại vị trí i, ta sẽ duyệt qua mọi ký tự d sao cho $d > S[i]$ và đếm số lần xuất hiện của ký tự d ở trước vị trí i.

- Sử dụng mảng cộng dồn hai chiều: $cnt[i][c]$ là số lần xuất hiện của ký tự c trong các vị trí j sao cho $0 \leq j \leq i$ ($0 \leq c < 26$, c đại diện cho 26 ký tự in thường trong bảng chữ cái từ 'a' đến 'z').
- Với mỗi vị trí i, ta duyệt qua tất cả mọi d ($S[i] < d < 26$), thực hiện `res += cnt[i][d]`.

Code mẫu:

```cpp
#include <bits/stdc++.h>
using namespace std;
const int k = 26;
const int maxN = 1e6 + 1;

int prefix_table[maxN][k];
string S;
int n;

void buildPrefixSum() {
    prefix_table[0][S[0] - 'a']++;

    for (int i = 1; i < n; ++i) {
        for (int j = 0; j < 26; ++j) {
            if (S[i] - 'a' == j) {
                prefix_table[i][j] = prefix_table[i - 1][j] + 1;
            } else {
                prefix_table[i][j] = prefix_table[i - 1][j];
            }
        }
    }
}

void solve() {
    long long ans = 0;
    for (int i = 0; i < n; ++i) {
        int currentChar = S[i] - 'a';
        for (int j = currentChar + 1; j < 26; ++j) {
            ans += prefix_table[i][j];
        }
    }
    cout << ans;
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    freopen("KHONGDEP.inp", "r", stdin);
    freopen("KHONGDEP.out", "w", stdout);
    cin >> n;
    cin >> S;
    buildPrefixSum();
    solve();
}
```

Bài 2. Mắt xích – MATXICH (3,5 điểm)

Subtask 1

Giới hạn: $n \leq 400$.

Xét tất cả n^2 xâu con $[l...r]$ của s, mỗi một xâu con ta duyệt qua trong O(n) từ l đến r để kiểm tra xâu con có phải là mắt xích không.
Độ phức tạp: $O(n^3)$.

Subtask 2

Giới hạn: $n \leq 2000$.
Ta vẫn duyệt trâu nhưng có sự cải tiến: từ trạng thái của $s[l...r]$, ta có thể kiểm tra cho $s[l...r+1]$ trong $O(1)$. Thật vậy, nếu $s[l...r]$ là mắt xích thì $s[l...r+1]$ là mắt xích khi $s[r+1] = s[r-1]$, còn nếu $s[l...r]$ không phải mắt xích thì chắc chắn $s[l...r+1]$ không phải mắt xích.
Độ phức tạp: $O(n^2)$.

Subtask 3

Yêu cầu kiến thức: 2 con trỏ.

- Nhận xét: Nếu $s[l...r]$ không là mắt xích thì $s[l...v]$ không là mắt xích với mọi $v \leq r$
- Từ đó, ta tưởng tượng 2 con trỏ l, r trong **Subtask 2**. Ban đầu ta cho `l = 1, r = 1` và tịnh tiến r. Nếu không nối dài được nữa (tức là nếu nối thêm thì đoạn hiện tại sẽ không còn là mắt xích), vì chắc chắn không còn tồn tại đoạn là mắt xích dài hơn bắt đầu tại l, ta dịch con trỏ l lên và tiếp tục quá trình trên.
- Một trong hai con trỏ phải dịch lên trong một lần chạy nên chỉ có tối đa $2n$ lần dịch lên và chạy.

Độ phức tạp: $O(n)$.

Code mẫu:

```cpp
#include <bits/stdc++.h>

#include <iostream>
#define ll long long
using namespace std;
const ll maxn = 5e5 + 5, INF = 4e18 + 9;
int n;
string s;
void inp() {
    cin >> n;
    cin >> s;
    s = ' ' + s;
}
void solve() {
    int i = 1, ans = 2;
    while (i <= n - 2) {
        int j = i + 1;
        while (j < n && s[j + 1] == s[j - 1]) {
            j++;
        }
        ans = max(ans, j - i + 1);
        i = j;
    }
    cout << ans;
}
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    freopen("MATXICH.inp", "r", stdin);
    freopen("MATXICH.out", "w", stdout);
    inp();
    solve();
    return 0;
}
```

Bài 3. Diện tích – AREA (3,0 điểm)

Subtask 1

Giới hạn: $n \leq 20$.

- Ta sẽ cố định góc trên bên trái và góc dưới bên phải của bảng con rồi duyệt lại bảng con xem nó chỉ chứa duy nhất một ký tự hay không.
- Duyệt góc trên bên trái và góc dưới bên phải mất độ phức tạp $O(N^4)$. Duyệt lại bảng con sẽ mất độ phức tạp $O(N^2)$ nên độ phức tạp cuối cùng của ta sẽ là $O(N^6)$. Với $N \leq 20$ thì dư sức chạy trong 1 giây.
- Việc xét một bảng con có nhiều cách xét. Trong code mẫu, ta sẽ xét số lớn nhất và số bé nhất trong bảng. Nếu số lớn nhất bằng số bé nhất thì bảng con đó chỉ bao gồm 1 ký tự.

Độ phức tạp: $O(N^6)$.

Code mẫu:

```cpp
#include <bits/stdc++.h>
using namespace std;
int arr[21][21];
int ans[21];
int n;
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen("AREA.INP", "r", stdin);
    freopen("AREA.OUT", "w", stdout);

    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> arr[i][j];
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            for (int u = i; u <= n; u++) {
                for (int v = j; v <= n; v++) {
                    int mn = n + 1, mx = 0;
                    for (int x = i; x <= u; x++) {
                        for (int y = j; y <= v; y++) {
                            mn = min(mn, arr[x][y]);
                            mx = max(mx, arr[x][y]);
                        }
                    }
                    if (mn == mx) {
                        ans[mn] = max(ans[mn], (u - i + 1) * (v - j + 1));
                    }
                }
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << ans[i] << ' ';
    }
}
```

Subtask 2

Giới hạn: $n \leq 100$.

Khi ta cố định số i và biến mảng ban đầu của ta thành bảng nhị phân (một phần tử có bằng i hay không). Khi đó, ta có thể cố định hai cột x,y là cạnh bên trái và bên phải của bảng con của ta. Sau đó, nếu ta sử dụng mảng cộng dồn, ta sẽ biết được rằng là hàng thứ u có chỉ toàn số i hay không. Nếu ta biểu diễn trạng thái của N dòng được giới hạn từ cột thứ x đến cột thứ y có toàn số i hay không là một mảng nhị phân khác thì ta sẽ phải chọn đoạn toàn 1 dài nhất. Việc này ta có thể làm trong $O(N)$.
Độ phức tạp: $O(N^4)$.

Code mẫu:

```cpp
#include <bits/stdc++.h>
using namespace std;
int arr[101][101];
int pref[101][101][101];
int ans[101];
int n;
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen("AREA.INP", "r", stdin);
    freopen("AREA.OUT", "w", stdout);

    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> arr[i][j];
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            for (int x = 1; x <= n; x++) {
                pref[x][i][j] = pref[x][i][j - 1];
            }
            pref[arr[i][j]][i][j]++;
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int x = 1; x <= n; x++) {
            for (int y = x; y <= n; y++) {
                bool flag = 0;
                int last = 0;
                for (int u = 1; u <= n; u++) {
                    if (pref[i][u][y] - pref[i][u][x - 1] == y - x + 1) {
                        if (flag == 0) {
                            last = u;
                            flag = 1;
                        }
                        ans[i] = max(ans[i], (u - last + 1) * (y - x + 1));
                    } else {
                        flag = 0;
                    }
                }
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << ans[i] << ' ';
    }
}
```

Subtask 3

Giới hạn: $n \leq 500$.

Gọi $H_{i,j}$ là số lớn nhất sao cho $A_{i,j} = A_{i-1,j} = A_{i-2,j} = ... = A_{i-H_{i,j}+1,j}$. Khi đó, khi ta xét dòng thứ u: Nếu ta cố định hai cạnh trái phải của hình chữ nhật côn là x và y thì ta dễ dàng tính diện tích lớn nhất dựa trên hàng bảng H. Ta tính mảng H trong $O(N^2)$. Ta cố định x và y và xét mỗi dòng sẽ mất $O(N^3)$.
Độ phức tạp: $O(N^3)$.

Code mẫu:

```cpp
#include <bits/stdc++.h>
using namespace std;
int arr[501][501];
int h[501][501];
int ans[501];
int n;
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen("AREA.INP", "r", stdin);
    freopen("AREA.OUT", "w", stdout);

    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> arr[i][j];
        }
    }
    for (int j = 1; j <= n; j++) {
        for (int i = 1; i <= n; i++) {
            if (arr[i][j] == arr[i - 1][j]) {
                h[i][j] = h[i - 1][j] + 1;
            } else {
                h[i][j] = 1;
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int x = 1; x <= n; x++) {
            int mn = h[i][x];
            for (int y = x; y <= n; y++) {
                if (arr[i][y] != arr[i][x]) break;
                mn = min(mn, h[i][y]);
                ans[arr[i][y]] = max(ans[arr[i][y]], mn * (y - x + 1));
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << ans[i] << ' ';
    }
}
```

— HẾT —