Lời giải [IT003.P21.CTTN.1] Assigment4 (Advance và Basic)

Bảo Quý Định Tân - 24520028

Ngày 12 tháng 5 năm 2025

Mục lục

1 Basic

1.1 [H007] - Alpha Problem

Vì $2 \leq x \leq 10$ nên ta thực hiện đổi số n từ hệ thập phân sang hệ x phân theo cách chia lấy dư và đảo ngược lại các số dư theo thứ tự xuất hiện. Ví dụ:

- $13/2 = 6$ dư 1
- $6/2 = 3$ dư 0
- $3/2 = 1$ dư 1
- $1/2 = 0$ dư 1

Ta viết ngược lại các số dư sẽ được dạng nhị phân của 13 là 1101.

```
vector<int> a;
while (n) {
    a.pb(n % x);
    n /= x;
}

reverse(all(a));
for (int x : a) {
    cout << x;
}
```

1.2 [H006] - Word Merging

Ta bắt đầu với nhận xét rằng không cần quan tâm thứ tự xóa như nào, cứ gặp 2 ký tự cạnh nhau giống nhau là ta cứ xóa thì đến cuối vẫn sẽ ra một kết quả giống nhau.

Cho nên ở đây ta tập trung việc thực hiện việc xoá như nào cho nhanh. Dựa trên ý tưởng giống như việc xoá dãy ngoặc hợp lệ, ta sử dụng thêm stack để hỗ trợ cài đặt.

```
string s;
cin >> s;

stack<char> st;
for (char h : s) {
    if (st.empty()) {
        st.push(h);
    } else {
```

```
        if (st.top() == h) {
            st.pop();
        } else {
            st.push(h);
        }
    }
}

cout << st.size();
```

1.3 [H004] - Wood Cutting

Thay vì nhìn bài toán theo hướng cắt các khúc gỗ ra, ta sẽ nhìn theo hướng ngược lại vì dễ giải hơn. Việc của ta là dán n khúc gỗ lại, với chi phí mỗi lần dán là tổng độ dài của 2 khúc gỗ.

Thì đây là một bài toán quen thuộc với hướng giải là mỗi lần dán, ta chỉ việc chọn 2 khúc gỗ có độ dài ngắn nhất để dán. Ta sử dụng *priority_queue* để hỗ trợ cài đặt thuận tiện hơn.

```
int n, s;
cin >> n >> s;

priority_queue<int, vector<int>, greater<int>> pq;
for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    pq.push(x);
}

int ans = 0;
while (pq.size() > 1) {
    int u = pq.top();
    pq.pop();
    int v = pq.top();
    pq.pop();
    ans += u + v;
    pq.push(u + v);
}

cout << ans;
```

1.4 [H003] - Isosceles Triangle

Bắt đầu bài toán, ta sẽ đếm số lượng cạnh tương ứng với từng độ dài.

```cpp
int n;
map<int, int> cnt;
cin >> n;
for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    cnt[x]++;
}
```

Ta cần đếm số lượng tam giác cân, có 3 cạnh là a, b, c. Giả sử $a = b$, thì điều kiện của c là $|a - b| < c < a + b$ hay $0 < c < 2 * a$.

Ta sẽ xét qua từng độ dài một theo thứ tự tăng dần, cố định độ dài hiện tại là cạnh a, việc còn lại là đếm số lượng cạnh c có thể ghép thành tam giác.

Để đếm số lượng $c < 2 * a$ nhanh ta có thể sử dụng ý tưởng hai con trỏ. Còn việc đếm số lượng tam giác có thể tạo ra được thì có 2 trường hợp:

Gọi $sum =$ số lượng $c < 2 * a$.

- TH1: $a = c$, ta gọi $cnt_a =$ số lượng cạnh a. Ta sẽ thêm vào đáp án $C(3, cnt_a) = \dfrac{cnt_a * (cnt_a - 1) * (cnt_a - 2)}{6}$.
- TH2: $a \neq c$, ta phải lấy riêng ra những số ta quan tâm sẽ là $sum - cnt_a = cnt$, ta sẽ thêm vào đáp án $C(2, cnt_a) * cnt = \dfrac{cnt_a * (cnt_a - 1)}{2} * cnt$.

```cpp
auto j = cnt.begin();
int sum = 0;
int ans = 0;
for (auto &it : cnt) {
    while (j != cnt.end() && j->fi < 2 * it.fi) {
        sum += j->se;
        j++;
    }
    sum -= it.se;
    ans += sum * (it.se - 1) * (it.se) / 2;
    sum += it.se;
    ans += it.se * (it.se - 1) * (it.se - 2) / 6;
}

cout << ans;
```

1.5 [H002] - Difference in Height

Để giải quyết vấn đề số lớn hơn hay bé hơn trong việc lấy trị tuyệt đối, ta sẽ sắp xếp mảng lại theo thứ tự tăng dần.

Xét số thứ i trong mảng sau khi sắp xếp, ta sẽ ghép cặp nó với các số $j < i$.

Vì $a_i >= a_j \ \forall \ j < i$ nên ta sẽ được tổng là $\sum_{j=1}^{i-1}(a_i - a_j) = a_i * (i-1) - \sum_{j=1}^{i-1} a_j$.

Nên ta sẽ duyệt qua mảng, duy trì một biến để lưu tổng các a_j và thực hiện cập nhật đáp án.

```
cin >> n;
for (int i = 1; i <= n; i++) {
    cin >> a[i];
}
sort(a + 1, a + n + 1);

int sum = 0;
int ans = 0;
for (int i = 1; i <= n; i++) {
    ans += a[i] * (i - 1) - sum;
    sum += a[i];
}

cout << ans;
```

1.6 [H001] - Tom's Currency

Để đếm ra được số lượng giá trị khác nhau, có nhiều cách nhưng một cách mình thường xài là nén số sử dụng vector.

```
int n;
cin >> n;
vector<int> a(n);
for (int &x : a) {
    cin >> x;
}
sort(all(a));
a.erase(unique(all(a)), a.end());
cout << (int) a.size() << '\n';
for (int x : a) {
    cout << x << ' ';
}
```

1.7 Hashing: VQ44-FLOWERS

Ở bài toán này, ta sẽ tham lam sử dụng các giá trị khác nhau trước tiên, nếu vẫn chưa hết K ta sẽ lấy đại các giá trị còn lại. Và để đếm số lượng của các giá trị, ta sử dụng *map*.

```cpp
vector<int> get_ans(const vector<int>& A,int K){
map<int,int> Hash;
vector<int> ans;

for (int x : A) {
    Hash[x]++;
}
for (auto &it : Hash) {
    if (K > 0) {
        ans.push_back(it.first);
        it.second--;
        K--;
    }
}
for (auto &it : Hash) {
    while (K > 0 && it.second > 0) {
        ans.push_back(it.first);
        it.second--;
        K--;
    }
}

return ans;
}
```

1.8 Hashing: KiemKe

Ở bài này ta cũng sử dụng *map* để đếm ra được số lượng giá trị khác nhau. Vì *map* lưu các giá trị theo cặp *key* và *val* tương ứng, nên số lượng *key* khác nhau, hay số lượng giá trị khác nhau cũng chính là kích thước của *map*.

```cpp
int count_distinct(const vector<string>& ids){
    map<string,int> Hash;

    for (auto it : ids) {
        Hash[it]++;
    }
    return Hash.size();
}
```