Nama : Muhammad Haekal Aditya Rahmadyan
NIM : 2602192071
Kelas : LD75

PROBLEM A

Analisis masalah ini berkaitan dengan penentuan kelayakan tiket emas dalam kompetisi INC 2023. Dalam permasalahan ini, terdapat N tim yang berpartisipasi, masing-masing diwakili oleh nama (Si) dan institusi terkait (Ti). Kriteria penentuan kelayakan mencakup tim-tim dari posisi M teratas yang langsung lolos, sementara tambahan Tiket Emas diberikan kepada paling banyak K tim dari institusi di luar M teratas. Penting untuk dicatat bahwa hanya tim dengan peringkat tertinggi dari setiap institusi yang memenuhi syarat untuk mendapatkan Tiket Emas.

Dalam hal input, terdapat tiga bilangan bulat, N, M, dan K, yang menyatakan jumlah total tim, jumlah tim teratas yang lolos secara langsung, dan jumlah maksimal Tiket Emas yang dapat diberikan. Tim-tim ini diidentifikasi melalui baris-baris input yang terdiri dari dua string, Si dan Ti, yang mewakili nama tim dan institusi masing-m asing.

Keluaran dari permasalahan ini melibatkan mencetak jumlah tim yang menerima Tiket Emas (X) dan nama-nama tim tersebut, diurutkan berdasarkan peringkat. Apabila tidak ada tim yang memenuhi syarat untuk Tiket Emas, maka nilai X akan 0, dan tidak ada nama yang dicetak.

Untuk memahami lebih lanjut, beberapa contoh skenario diberikan. Misalnya, jika terdapat satu institusi yang memenuhi syarat untuk Tiket Emas, seperti UHOLO dengan tim NOIHS di peringkat 2 pada contoh pertama, maka hasil output akan mencantumkan jumlah tim yang mendapatkan Tiket Emas (X) beserta nama-nama tim tersebut, sesuai dengan format yang dijelaskan.

Pendekatan solusi yang diusulkan melibatkan iterasi melalui tim-tim untuk menemukan tim dengan peringkat tertinggi dari setiap institusi. Setelah itu, institusi teratas M diidentifikasi, bersama dengan institusi tambahan yang memenuhi syarat untuk Tiket Emas. Tim-tim yang memenuhi syarat tersebut kemudian diurutkan berdasarkan peringkat mereka untuk mencapai hasil akhir yang diharapkan.

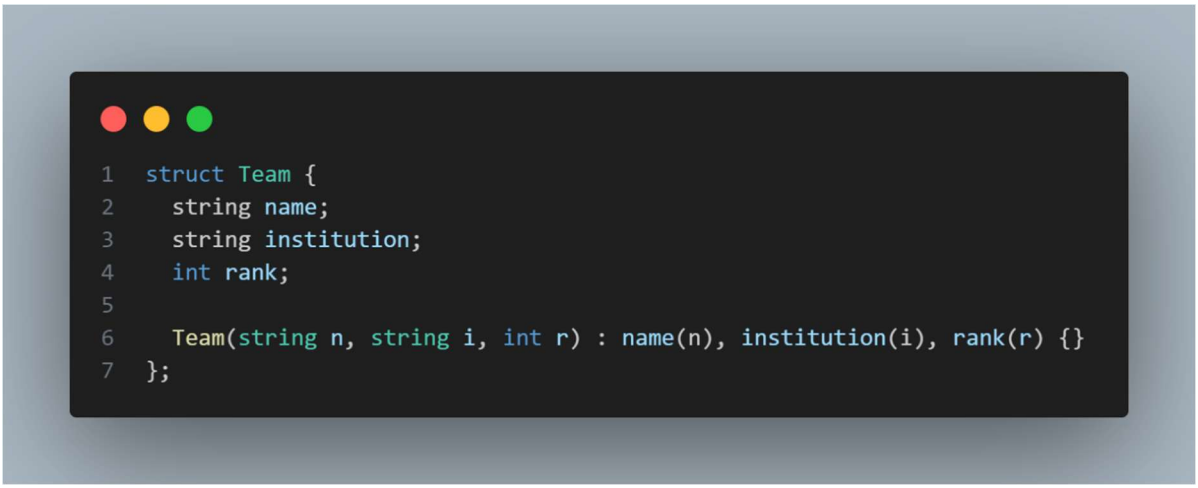
Dengan demikian, kompleksitas waktu dari pendekatan ini adalah O(NlogN), yang dipengaruhi oleh proses pengurutan tim. Sementara itu, kompleksitas ruangnya adalah O(N), karena diperlukan penyimpanan informasi mengenai tim dengan peringkat tertinggi dari setiap institusi.

Berikut merupakan Code penyelesaian dari problem diatas

```cpp
#include <bits/stdc++.h>
#define ll long long
#define INF 1000000000000000003
#define MOD 1000000007

#define f(i, a, b) for (int i = a; i < b; i++)
#define fz(i, a) f(i, 0, a)
#define ffz(i, j, a, b) f(i, 0, a) f(j, 0, b)

#define v(name, type) vector<type> name;
#define vi(name, type, n) vector<type> name(n, 0);
#define vin(name, n) fz(i, n) cin >> name[i];
#define vout(name, n) fz(i, n) cout << name[i] << " ";

using namespace std;

struct Team {
  string name;
  string institution;
  int rank;

  Team(string n, string i, int r) : name(n), institution(i), rank(r) {}
};

bool compareTeams(const Team &a, const Team &b) { return a.rank < b.rank; }

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  int N, M, K;
  cin >> N >> M >> K;

  v(teams, Team);
  map<string, int> institutionRanks;

  fz(i, N) {
    string name, institution;
    cin >> name >> institution;
    teams[i] = Team(name, institution, i + 1);

    if (institutionRanks.find(institution) == institutionRanks.end()) {
      institutionRanks[institution] = i + 1;
    }
  }

  sort(teams.begin(), teams.end(), compareTeams);

  int goldenTicketCount = 0;
  vector<string> goldenTicketTeams;

  for (const Team &team : teams) {
    if (goldenTicketCount >= K) {
      break;
    }

    if (team.rank > M && team.rank == institutionRanks[team.institution]) {
      goldenTicketTeams.push_back(team.name);
      goldenTicketCount++;
    }
  }

  cout << goldenTicketCount << endl;
  for (const string &team : goldenTicketTeams) {
    cout << team << endl;
  }

  return 0;
}
```

Penjelasan Code :

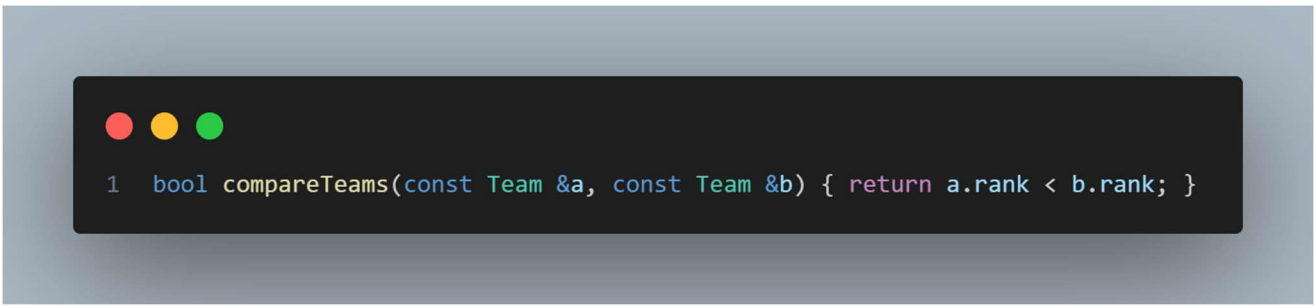
1. Struct Team



```
struct Team {
  string name;
  string institution;
  int rank;

  Team(string n, string i, int r) : name(n), institution(i), rank(r) {}
};
```

Penjelasan: Ini adalah definisi struktur **Team**. Struktur ini memiliki tiga atribut: **name** (nama tim), **institution** (institusi asal tim), dan **rank** (peringkat tim). Konstruktor digunakan untuk inisialisasi objek **Team** dengan nilai-nilai awal.

2. compareTeams Function



```
bool compareTeams(const Team &a, const Team &b) { return a.rank < b.rank; }
```

Penjelasan: Ini adalah fungsi perbandingan untuk digunakan dalam pengurutan vektor **teams**. Fungsi ini membandingkan dua objek **Team** berdasarkan peringkatnya.

3. Membaca Informasi Tim dan Membuat Map Institution Ranks

```
fz(i, N) {
    string name, institution;
    cin >> name >> institution;
    teams[i] = Team(name, institution, i + 1);

    if (institutionRanks.find(institution) == institutionRanks.end()) {
      institutionRanks[institution] = i + 1;
    }
  }
```

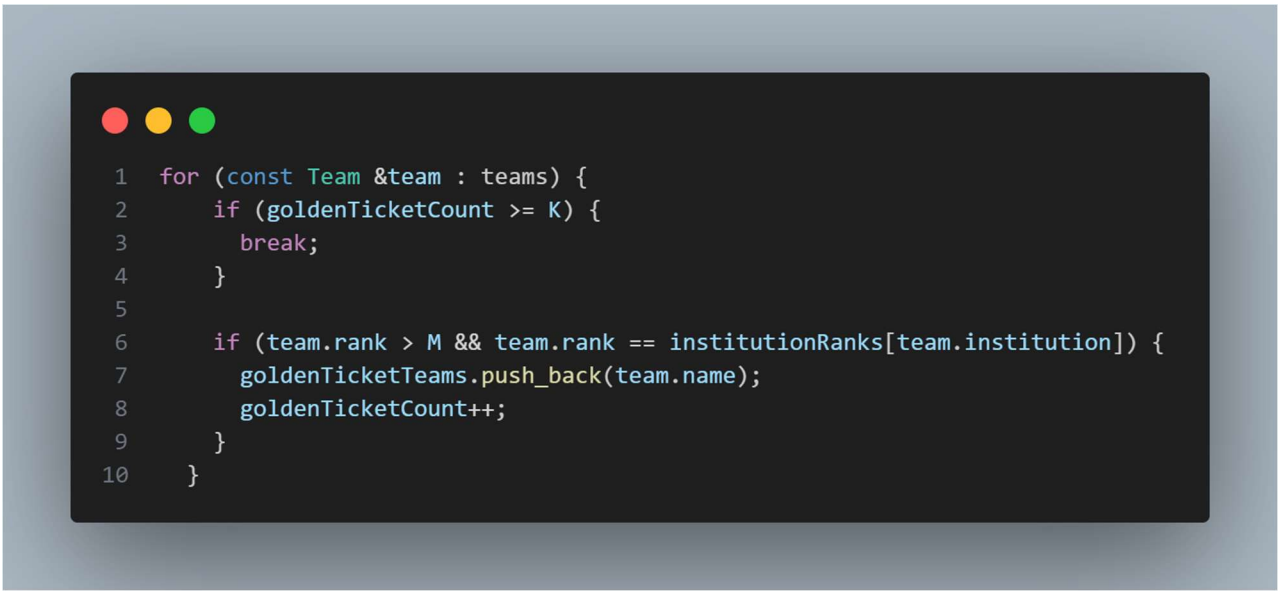
Penjelasan: Ini adalah loop untuk membaca informasi tim dan menginisialisasi vektor **teams**. Selain itu, program memeriksa apakah institusi sudah ada di **institutionRanks**, dan jika belum, maka peringkat terendah dari institusi tersebut ditambahkan ke peta.

4. Sorting Team

```
1  sort(teams.begin(), teams.end(), compareTeams);
```

Penjelasan: Tim-tim diurutkan berdasarkan peringkatnya menggunakan fungsi **compareTeams**. Ini memastikan bahwa vektor **teams** sekarang berisi tim-tim yang telah diurutkan berdasarkan peringkatnya.

5. Seleksi Pemenang Tiket Emas



```
1  for (const Team &team : teams) {
2      if (goldenTicketCount >= K) {
3        break;
4      }
5
6      if (team.rank > M && team.rank == institutionRanks[team.institution]) {
7        goldenTicketTeams.push_back(team.name);
8        goldenTicketCount++;
9      }
10   }
```

Penjelasan: Ini adalah loop untuk seleksi pemenang tiket emas. Program memeriksa apakah tim memenuhi kriteria untuk memenangkan tiket emas. Jika iya, nama tim ditambahkan ke vektor **goldenTicketTeams**

PROBLEM B

Permasalahan ini adalah tentang membuat rencana diet untuk N hari ke depan (dari hari 1 hingga N). Selama setiap hari i, Anda harus minum tepat Pi mL susu. Alternatifnya, Anda bisa mengonsumsi sebanyak mungkin biskuit sebagai pengganti susu pada hari itu.

Anda memiliki persediaan M mL susu dan K biskuit. Jika tidak cukup susu untuk diminum pada suatu hari dan Anda kehabisan biskuit, maka rencana diet Anda berakhir. Tujuan dari permasalahan ini adalah menentukan jumlah maksimum hari yang dapat Anda pertahankan dalam rencana diet Anda dengan persediaan susu dan biskuit yang diberikan.

Input pertama berisi tiga bilangan bulat N, M, dan K, yang mewakili jumlah hari dalam rencana diet, jumlah mL susu yang dimiliki, dan jumlah biskuit yang dimiliki. Input kedua berisi N bilangan bulat Pi, yang mewakili jumlah susu yang harus diminum setiap hari.

Output yang diharapkan adalah satu bilangan bulat yang menyatakan jumlah maksimum hari yang dapat Anda pertahankan dalam rencana diet. Berikut merupakan penyelesaian

```cpp
#include <bits/stdc++.h>
#define ll long long
#define INF 1000000000000000003
#define MOD 1e9 + 7

#define f(i, a, b) for (int i = a; i < b; i++)
#define fz(i, a) f(i, 0, a)
#define ffz(i, j, a, b) f(i, 0, a) f(j, 0, b)

#define vi(name, type, n) vector<type> name(n, 0);
#define vin(name, n) fz(i, n) cin >> name[i];
#define vout(name, n) fz(i, n) cout << name[i] << " ";

using namespace std;

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  ll N, M, K;
  cin >> N >> M >> K;

  vi(P, ll, N);
  vin(P, N);

  priority_queue<ll> pq;
  int d = 0;
  fz(i, N) {
    pq.push(P[i]);
    if (M < P[i]) {
      if (K == 0) break;
      K--;
      M += pq.top();
      pq.pop();
    }
    M -= P[i];
    d++;
  }

  cout << d << endl;

  return 0;
}
```

Penjelasan code

1. Deklarasi dan Pendefinisian Makro:
    - ll: Alias untuk long long.
    - INF: Konstanta untuk representasi nilai tak terhingga.
    - MOD: Konstanta untuk representasi modulus.
2. Makro dan Fungsi Iterasi:
    - f(i, a, b): Makro untuk melakukan iterasi dari a hingga b.
    - fz(i, a): Makro untuk melakukan iterasi dari 0 hingga a.
    - ffz(i, j, a, b): Makro untuk melakukan nested iteration.
3. Deklarasi Vector dan Input:
    - vi(name, type, n): Makro untuk mendeklarasikan vector dari tipe type dengan nama name dan ukuran n.
    - vin(name, n): Input nilai-nilai vector name sebanyak n.
    - vout(name, n): Output nilai-nilai vector name sebanyak n.
4. Input Utama:
    - Membaca nilai N, M, dan K dari input.
    - Membaca nilai-nilai array P sebanyak N.
5. Penggunaan Priority Queue:
    - Membuat priority queue (pq) untuk menyimpan nilai-nilai array P dalam urutan menurun.
    - Iterasi melalui setiap hari dalam rencana diet (array P).
    - Menambahkan nilai P[i] ke dalam priority queue.
    - Jika M kurang dari P[i], maka:
        - Jika masih ada biskuit (K > 0), konsumsi biskuit tersebut.
        - Tambahkan nilai terbesar dari priority queue ke M.
        - Hapus nilai terbesar dari priority queue.
    - Kurangkan M dengan P[i].
    - Tambahkan 1 pada variabel d yang menyimpan jumlah hari bertahan.

PROBLEM H

Masalah ini melibatkan penemuan gua yang penuh dengan N harta karun (diberi nomor dari 1 hingga N). Setiap harta karun i memiliki berat Wi dan nilai Vi. Untungnya, membawa M kereta kuda (diberi nomor dari 1 hingga M) untuk membantu mengangkut harta karun. Setiap kereta hanya dapat membawa satu harta karun; kereta j hanya dapat membawa harta karun dengan berat paling banyak Sj.

Tujuan dari masalah ini adalah untuk menentukan nilai total maksimum dari harta karun yang dapat dibawa menggunakan kereta kuda yang terbatas. Setiap harta karun memiliki berat dan nilai, dan Anda harus memilih cara menempatkan harta karun tersebut ke dalam kereta kuda Anda sehingga nilai totalnya maksimal, memperhatikan batasan berat setiap kereta.

Input dari masalah ini terdiri dari beberapa bagian:

- Baris pertama berisi dua bilangan bulat N dan M ($1 \leq N, M \leq 100,000$), mewakili jumlah harta karun dan jumlah kereta kuda.
- N baris berikutnya masing-masing berisi dua bilangan bulat Wi dan Vi ($1 \leq Wi, Vi \leq 1,000,000$), mewakili berat dan nilai dari setiap harta karun.
- Baris selanjutnya berisi M bilangan bulat Sj ($1 \leq Sj \leq 1,000,000$), mewakili kapasitas berat dari masing-masing kereta kuda.

Output dari masalah ini berupa satu bilangan bulat, yang merupakan nilai total maksimum dari harta karun yang dapat Anda bawa menggunakan kereta kuda.

Selanjutnya, terdapat beberapa contoh input dan output yang diberikan untuk menjelaskan cara masalah ini harus diselesaikan. Pada contoh-contoh tersebut, dijelaskan langkah-langkah atau cara penempatan harta karun ke dalam kereta kuda yang menghasilkan nilai total maksimum.

Masalah ini dapat diselesaikan dengan menggunakan pendekatan pemrograman dinamis, di mana nilai maksimum dari harta karun yang dapat diangkut dihitung secara iteratif berdasarkan nilai optimal dari submasalah yang lebih kecil. Solusi optimal dapat ditemukan dengan mempertimbangkan batasan berat masing-masing kereta kuda.

Berikut Code penyelesaian masalah

```cpp
#include <bits/stdc++.h>
#define ll long long
#define INF 1000000000000000003
#define MOD 1e9 + 7

#define f(i, a, b) for (int i = a; i < b; i++)
#define fz(i, a) f(i, 0, a)
#define ffz(i, j, a, b) f(i, 0, a) f(j, 0, b)

#define vi(name, type, n) vector<type> name(n, 0);
#define vin(name, n) fz(i, n) cin >> name[i];
#define vout(name, n) fz(i, n) cout << name[i] << " ";

using namespace std;

void solve() {
  ll N, M, total = 0;
  cin >> N >> M;

  vector<pair<ll, ll>> treasures(N);
  fz(i, N) cin >> treasures[i].first >> treasures[i].second;

  vi(carts, ll, M);
  vin(carts, M);

  sort(treasures.begin(), treasures.end());
  sort(carts.begin(), carts.end());

  priority_queue<int> pq;
  int j = 0;

  for (int i = 0; i < M; i++) {
    while (j < N && treasures[j].first <= carts[i]) {
      pq.push(treasures[j].second);
      j++;
    }
    if (!pq.empty()) {
      total += pq.top();
      pq.pop();
    }
  }

  cout << total << "\n";
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  solve();

  return 0;
}
```

Berikut penjelasan function main ditandai dengan comment

```cpp
void solve() {
  ll N, M, total = 0;
  cin >> N >> M;

  // Membuat vektor pasangan untuk menyimpan berat dan nilai setiap harta karun
  vector<pair<ll, ll>> treasures(N);
  fz(i, N) cin >> treasures[i].first >> treasures[i].second;

  // Membuat vektor untuk menyimpan kapasitas berat setiap kereta kuda
  vi(carts, ll, M);
  vin(carts, M);

  // Mengurutkan vektor harta karun berdasarkan berat dan vektor kapasitas berat kereta kuda
  sort(treasures.begin(), treasures.end());
  sort(carts.begin(), carts.end());

  // Menggunakan priority queue untuk menyimpan nilai harta karun yang sesuai dengan kapasitas berat kereta kuda
  priority_queue<int> pq;
  int j = 0;

  // Iterasi melalui setiap kereta kuda
  for (int i = 0; i < M; i++) {
    // Memasukkan harta karun ke dalam priority queue berdasarkan berat
    while (j < N && treasures[j].first <= carts[i]) {
      pq.push(treasures[j].second);
      j++;
    }

    // Jika priority queue tidak kosong, mengambil harta karun dengan nilai tertinggi
    if (!pq.empty()) {
      total += pq.top();
      pq.pop();
    }
  }

  // Mencetak total nilai maksimum dari harta karun yang dapat diangkut
  cout << total << "\n";
}
```