KPP Programming Pelatdas 2025

SOAL

Setelah mengikuti pelatihan dasar divisi programming dari ***UKM Robotika, kalian merasa siap dalam menyelesaikan masalah apapun terkait pemrograman. Sebagai tes terakhir, kalian diminta untuk membuat robot asisten yang dapat membantu pekerjaan para anggota tim robotik kesukaan kalian.***

Setelah pelatihan, tim robotika membuat robot otonom untuk menavigasi jaringan jalan. Selain panjang jalan biasa, beberapa jalan memiliki gundukan (obstacle) yang menambah biaya energi saat dilewati. Robot harus menemukan rute dari S ke T yang meminimalkan total energi yang dikonsumsi, sambil memperhitungkan waktu (pengaruh ganjil/genap), kemampuan menunggu di rest point, dan mengisi ulang di charging station.

Aturan penting

1. Graf: N node, M edge. Setiap edge diberi dua nilai:
   - w = panjang dasar (meter).
   - o = bobot obstacle (meter tambahan) — bisa 0 jika tidak ada gundukan.
   - Total energi dasar untuk melewati edge = w + o.
2. Pengaruh waktu:
   - Jika edge dilewati pada jam ganjil → energi untuk edge bertambah 30% (kalikan 1.3).
   - Jika edge dilewati pada jam genap → energi untuk edge berkurang 20% (kalikan 0.8).
   - Waktu di sini dihitung dalam menit; setiap langkah (melewati satu edge) menghabiskan waktu sesuai kecepatan saat itu (lihat poin 4).
3. Rest point (R): robot boleh menunggu integer menit di node ini untuk mengganti jam ganjil/genap sebelum melanjutkan. Menunggu boleh berulang kali.
4. Charging station (C): saat berada di node C, robot bisa mengisi ulang energi ke nilai maksimum (1000 meter).
5. Batas energi: energi maksimum = 1000 meter. Robot memulai penuh. Jika pada titik mana pun tidak cukup energi untuk melewati edge dan tidak ada charging station yang bisa dicapai sebelum energi habis → perjalanan gagal.
6. Tujuan: cari rute dari S ke T yang meminimalkan total energi yang dikonsumsi (energ
i yang dikonsumsi dihitung sebagai penjumlahan energi sesungguhnya untuk setiap edge ketika dilewati — termasuk efek waktu dan obstacle). Output juga harus menyertakan jalur dan waktu tiba di setiap node pada jalur tersebut.

Format masukan:

- Baris pertama: N M (jumlah node, jumlah edge).
- Berikutnya M baris: u v w o (edge tak-berarah antara node u dan v, panjang dasar w, obstacle o integer ≥ 0).
- Baris berikutnya: S T (nama node start dan target).
- Baris selanjutnya: daftar node rest point dipisah spasi (atau - jika tidak ada).
- Baris selanjutnya: daftar node charging station dipisah spasi (atau - jika tidak ada).
- Baris selanjutnya: node M jika ada (mechanic) atau -,
- Baris selanjutnya: node E jika ada (electrical) atau -.
- Baris terakhir: jam awal perjalanan (integer jam, mis. 1 artinya jam ganjil).

Format Output:

Jika berhasil:

- Total energi minimum: [nilai_integer]
- Jalur: S -> . . . -> T
- Waktu tiba: lalu setiap baris Node (jam X) sesuai waktu saat mencapai node itu (jam dihitung sebagai integer menit dibagi 1 menit per unit waktu? — gunakan menit sebagai satuan lalu konversi ke jam integer sesuai soal; untuk konsistensi: gunakan menit absolut dan tampilkan menit, contoh S (menit 0), atau kalau mau jam gunakan jam mulai + floor(minit/60). (Contoh format:S (menit 0), A (menit 2)`, dst.)

Jika gagal karena nabrak/dinding tak bisa dilewati atau energi habis dan tidak ada charging sebelum itu:

Robot gagal dalam mencapai tujuan :(

Batasan:

- N ≤ 200, M ≤ 2000.
- 0 ≤ w, o ≤ 500.
- Energi maksimum = 1000 meter.
- Jumlah rest point ≤ 10; jumlah charging station ≤ 10.

Sample Input:
N x M: 6 7 ====PLAY==== S A 200 0 A B 300 50 A C 150 0 B D 400 100 C D 100 0 D E 250 0 E T 200 0 S T

C E M - 1
Sample Output:
Total energi minimum: 960 Jalur: S -> A -> C -> D -> E -> T Waktu tiba: S (menit 0) A (menit 2) C (menit 4) D (menit 6) E (menit 8) T (menit 10)

Sample input 2:
4 3 S A 600 200 A B 300 0 B T 200 0 S T - - - - 1
Sample output 2:
Robot gagal dalam mencapai tujuan :(

Nilai plus apabila:

- Menerapkan OOP pada program
- Menggunakan algoritma path finding

Penyelesaian

Bagus — aku ringkas langkah-langkah praktis (nggak terlalu kaku) supaya gampang kamu masukkan ke README GitHub atau catatan pengerjaan. Tiap bagian singkat, langsung ke poin penting.

Pemahaman Masalah

1. Tujuan: dari node `S` ke node `T` cari rute yang **meminimalkan total energi** yang dikonsumsi.
2. Constraint penting:
    - Edge punya dua nilai: `w` (panjang) dan `o` (obstacle). Energi dasar per edge = `w + o`.
    - Energi terpakai tergantung *jam* saat melewati edge:
        - jam ganjil → ×1.3, jam genap → ×0.8.
    - Waktu diukur dalam menit; start punya `StartHour`.
    - Time to traverse edge = `ceil(w / speed)` (gunakan `w` untuk waktu, `w+o` untuk energi).
    - Node `R` = restpoint (boleh/harus tunggu), `C` = charging (isi ulang ke 1000).
    - Energi maksimal = 1000. Jika tidak cukup energi dan tidak ada charging reachable → gagal.
3. Output: total energi minimal, jalur (nama node), dan waktu tiba setiap node (menit / jam ke-berapa).

Perumusan Model

- Representasi:
    - Graf tak-berarah G=(V,E)G=(V,E) dengan edge beratribut `{w, o}`.
    - Map `name -> id` (string nodes).
- Variabel waktu/energi:
    - `timeNow` = menit sejak start.
    - `hour_now = StartHour + floor(timeNow / 60)` → digunakan untuk paritas.
    - `timeEdge = ceil(w / speed)` (menit).
    - `energyCost = ceil((w + o) * factor)` where `factor = 1.3` if `hour_now` ganjil else `0.8`.
- Restpoint rule (kebijakan implementasi kita): jika tiba di `R` saat `hour_now` ganjil → tunggu sampai jam berikutnya genap:
    - `delay = 60 - (timeNow % 60)`; `timeNow += delay`.
- Charging: jika berada di `C`, set `energyLeft = 1000` sebelum melanjutkan.

Algoritma yang Digunakan

- Gunakan **Dijkstra yang dimodifikasi (state-augmented)** — priority queue berbasiskan `energyUsed` (kita ingin minimalkan energi terpakai).

State yang disimpan di PQ:

(energyUsed, nodeId, timeNow, energyLeft, path(vector<id>), times(vector<timeNow>))

- menyimpan `path`+`times` supaya output waktu sinkron.
- **Transisi** saat pop state:
    1. Jika `node == target` → selesai, print `path` dan `times`.
    2. Jika di `R` dan `hour` ganjil → apply delay (update `timeNow`).

3. Untuk setiap tetangga `v`:
    - `newTime = timeNow + ceil(w / speed)`
    - tentukan `factor` dari `hour = StartHour + floor(timeNow/60)` (pakai hour sebelum bergerak)
    - `cost = ceil((w+o) * factor)`
    - jika `node` adalah `C` → `energyLeft = 1000` (recharge) sebelum cek cost
    - jika `energyLeft < cost` → skip
    - `newEnergyLeft = energyLeft - cost`, `newEnergyUsed = energyUsed + cost`
    - push state baru ke PQ

- **Pruning**: table `best[node][energyLeft]` menyimpan energiUsed minimal yang pernah dicapai; kalau `newEnergyUsed >= best[nxt][newEnergyLeft]` → skip.
- Stop condition: PQ empty → gagal.

Desain Data Struktur

- `unordered_map<string,int> nameToId; vector<string> idToName;` — mapping string↔id.
- `struct Edge { int to; int w; int o; };` atau `pair<int, pair<w,o>>` but lebih jelas pakai struct.
- `vector<vector<Edge>> adj;` — adjacency list.
- `vector<char> nodeRole` ('.' default, 'R','C','M','E' bila ada).
- `const int EMAX = 1000;`
- `vector<vector<int>> best` atau `vector<unordered_map<int,int>> bestPerNode` untuk memperkecil memori (jika N kecil, 256×1001 ok).

- PQ: `priority_queue< State, vector<State>, greater<State> >` dengan `State` = `tuple<int,int,int,int,vector<int>,vector<int>>`.
- Fungsi util:
    - `int getNodeId(string)` — cek batas `N` dan buat id baru.
    - Input parsing: gunakan `cin >>` untuk token, tapi `cin.ignore()` sebelum `getline()`.

Implementasi dalam C++ (outline + potongan kunci)

Source Code CPP

Contoh Input dan Output

Sample Input
6 7 S A 200 0 A B 300 50 A C 150 0 B D 400 100 C D 100 0 D E 250 0 E T 200 0 S T C E M - 1
Sample Output
Total energi minimum: 895 Jalur: S -> A -> C -> D -> E -> T Waktu tiba: S (Jam 1 Menit 0) A (Jam 1 Menit 2) C (Jam 1 Menit 4) D (Jam 2 Menit 1) E (Jam 2 Menit 4)

T (Jam 2 Menit 6)

Sample Input
4 3 S A 600 200 A B 300 0 B T 200 0 S T - - - - 1
Sample Output
Robot gagal dalam mencapai tujuan :(