KPP Programming Pelatdas 2025

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