# DRAFT JAWABAN UTS: STUDI KASUS ROBOT PEMANDU KAMPUS

**Mata Kuliah:** Kecerdasan Buatan (Artificial Intelligence)

## 💡 Pendahuluan: Konsep "Smart Campus Navigation"

Sebelum masuk ke jawaban teknis, kita mengasumsikan bahwa robot ini (kita beri nama **"RoboGuide"**) tidak hanya sekadar menghitung angka, tetapi mensimulasikan cara kerja *Google Maps* atau *Waze* dalam skala mikro (kampus). Masalah ini adalah masalah **Search & Optimization** klasik dalam AI.

## ✅ SOAL 1: Representasi Ruang Keadaan (State Space)

**(Bobot: 20%)**

**Analisis:** Masalah navigasi kampus ini dapat dimodelkan sebagai **Weighted Undirected Graph** (Graf Tak Berarah Berbobot).

- **Node (Simpul):** Gedung/Lokasi ($\alpha, \Omega, \theta, \mu, KI, \lambda, O, I, E, K, \Sigma, \tau$).
- **Edge (Sisi):** Jalur yang menghubungkan antar gedung.
- **Cost (Bobot):** Waktu tempuh dalam menit.

**Jawaban Visual (Grafik):** Berikut adalah representasi graf berdasarkan data keterhubungan (adjacency) yang diberikan:

```
graph TD
    %% Definisi Node dan Koneksi dengan Bobot (Menit)
    Alpha((Alpha α)) ---|3| Omega((Omega Ω))
    Alpha ---|7| Mu((Mu μ))
    Alpha ---|3| Theta((Theta θ))

    Omega ---|7| Mu
    Omega ---|2| KI((Kamoike KI))

    Theta ---|5| Mu
    Theta ---|2| Lambda((Lambda λ))

    Mu ---|6| KI
    Mu ---|6| Lambda
    Mu ---|8| Omicron((Omicron O))
    Mu ---|8| Iota((Iota I))
    Mu ---|8| Epsilon((Epsilon E))

    Lambda ---|4| Omicron

    Omicron ---|2| Iota
    Omicron ---|5| Tau((Tau τ))

    Iota ---|2| Epsilon

    Epsilon ---|2| Kappa((Kappa K))

    Kappa ---|3| Sigma((Sigma Σ))
```

*Catatan untuk Mahasiswa: Jika Anda menggambar manual di kertas, pastikan Gedung Mu ($\mu$) diletakkan di tengah karena memiliki koneksi terbanyak (central hub), sehingga grafik terlihat rapi dan tidak terlalu banyak garis yang bertabrakan.*

## ✅ SOAL 2: Pencarian Jalur (Blind Search)

**(Bobot: 40%)**

**Skenario:**

- **Start State:** Gedung Omega ($\Omega$)
- **Goal State:** Gedung Sigma ($\Sigma$)

### A. Breadth-First Search (BFS)

**Konsep:** RoboGuide akan mengecek semua gedung tetangga terdekat terlebih dahulu (seperti menyebarkan radar ke segala arah) sebelum bergerak lebih jauh. BFS menjamin solusi ditemukan dengan langkah (hop) paling sedikit, meski belum tentu waktu tercepat.

**Proses Pencarian (Open List / Antrian):**

1. **Level 0:** Buka $\Omega$.
2. **Level 1:** Cek tetangga $\Omega \rightarrow \{\alpha, \mu, KI\}$. (Belum ada $\Sigma$, lanjut).
3. **Level 2:**
   - Dari $\alpha$ cek $\{\theta\}$.
   - Dari $\mu$ cek $\{\lambda, O, I, E\}$.
   - Dari $KI$ (Buntu/kembali ke node yang sudah dikunjungi).
4. **Level 3:**
   - Dari $\theta$ cek $\{\lambda\}$ (sudah ada di antrian).
   - Dari $E$ cek $\{K\}$.
5. **Level 4:**
   - Dari $K$ cek $\{\Sigma\}$. **(GOAL DITEMUKAN!)**

**Urutan Kunjungan (Traversal Order):**
$\Omega \rightarrow \alpha \rightarrow \mu \rightarrow KI \rightarrow \theta \rightarrow \lambda \rightarrow O \rightarrow I \rightarrow E \rightarrow K \rightarrow \Sigma$

**Rute Solusi BFS:** $\Omega \rightarrow \mu \rightarrow E \rightarrow K \rightarrow \Sigma$

### B. Depth-First Search (DFS)

**Konsep:** RoboGuide akan mencoba menelusuri satu jalur sedalam mungkin sampai mentok (dead-end) atau ketemu tujuan, baru *backtracking* (mundur) jika salah. Ini seperti orang yang nekad mencoba satu jalan lurus terus.

**Asumsi:** Kita memprioritaskan node tetangga berdasarkan urutan abjad atau urutan data tabel untuk konsistensi.

**Pohon Pencarian & Jejak (Trace):**

1. Mulai $\Omega$.

2. Pilih cabang ke $\alpha$.
3. Dari $\alpha$, pilih cabang ke $\theta$.
4. Dari $\theta$, pilih cabang ke $\lambda$.
5. Dari $\lambda$, pilih cabang ke $O$.
6. Dari $O$, pilih cabang ke $I$ (misal prioritas ke Iota sebelum Tau).
7. Dari $I$, pilih cabang ke $E$.
8. Dari $E$, pilih cabang ke $K$.
9. Dari $K$, pilih ke $\Sigma$. **(GOAL)**

*Catatan: DFS sangat bergantung pada urutan pemilihan node. Jika robot "beruntung" memilih jalur $\mu$ duluan, rutenya bisa berbeda.*

**Urutan Kunjungan:** $\Omega \rightarrow \alpha \rightarrow \theta \rightarrow \lambda \rightarrow O \rightarrow I \rightarrow E \rightarrow K \rightarrow \Sigma$

## ✅ SOAL 3: Heuristic Search (Pencarian Cerdas)

**(Bobot: 40%)**

**Analisis Kritis (Out of The Box):** Soal meminta "Algoritma Pencarian Heuristik" (biasanya A* atau Greedy Best First). Namun, data soal **tidak menyediakan nilai Heuristik ($h(n)$)** atau jarak garis lurus (Straight Line Distance) dari setiap gedung ke tujuan.

Dalam kasus nyata AI, jika data heuristik tidak tersedia, algoritma terbaik yang bisa digunakan adalah **Uniform Cost Search (UCS)** atau algoritma **Dijkstra**. Algoritma ini dianggap "optimal" karena menjamin rute dengan **biaya (cost) terendah**, bukan hanya jumlah langkah tersedikit.

**Konversi Satuan:**

- 1 menit = 60 meter.

### Tabel Solusi & Perhitungan Rute Tercepat

Berikut adalah hasil pelacakan menggunakan prinsip *Lowest Path Cost* ($g(n)$):

| Rute (Start ke Goal) | Rute Tercepat (Jalur Node) | Perhitungan Waktu (Menit) | Jarak Tempuh (Meter)<br>*(Waktu x 60)* |
|---|---|---|---|
| **Alpha ($\alpha$) ke Tau ($\tau$)** | $\alpha \rightarrow \theta \rightarrow \lambda \rightarrow O \rightarrow \tau$ | $3 + 2 + 4 + 5 = \mathbf{14}$ | $14 \times 60 = \mathbf{840}$ m |
| **Omega ($\Omega$) ke Sigma ($\Sigma$)** | $\Omega \rightarrow \mu \rightarrow E \rightarrow K \rightarrow \Sigma$ | $7 + 8 + 2 + 3 = \mathbf{20}$ | $20 \times 60 = \mathbf{1.200}$ m |
| **Lambda ($\lambda$) ke Kappa ($K$)** | $\lambda \rightarrow O \rightarrow I \rightarrow E \rightarrow K$ | $4 + 2 + 2 + 2 = \mathbf{10}$ | $10 \times 60 = \mathbf{600}$ m |
| **Theta ($\theta$) ke Omicron ($O$)** | $\theta \rightarrow \lambda \rightarrow O$ | $2 + 4 = \mathbf{6}$ | $6 \times 60 = \mathbf{360}$ m |

**Penjelasan Detail Perhitungan (Justifikasi untuk Dosen)**

**1. Alpha ke Tau ($\alpha \to \tau$)**

- Opsi 1 (via Mu): $\alpha \to \mu \to O \to \tau = 7 + 8 + 5 = 20$ menit.
- Opsi 2 (via Theta): $\alpha \to \theta \to \lambda \to O \to \tau = 3 + 2 + 4 + 5 = 14$ menit.
- **Kesimpulan:** Jalur via Theta lebih efisien karena menghindari kemacetan/jarak jauh di gedung $\mu$.

**2. Omega ke Sigma ($\Omega \to \Sigma$)**

- Opsi 1 (via Alpha): $\Omega \to \alpha \to \cdots \to \Sigma$ (total akan > 20 menit).
- Opsi 2 (via Mu): $\Omega \to \mu(7) \to E(8) \to K(2) \to \Sigma(3)$ = 20 menit.
- *Catatan:* Jalur via Mu terlihat "mahal" di awal (7 menit), tapi memotong jalan pintas langsung ke sayap kanan kampus (Epsilon).

**3. Lambda ke Kappa ($\lambda \to K$)**

- Mengapa tidak lewat Mu? $\lambda \to \mu(6) \to E(8) \ldots$ sudah 14 menit.
- Lewat Omicron lebih cepat: $\lambda \to O(4) \to I(2) \to E(2) \to K(2)$ = 10 menit. Ini contoh di mana "banyak hops" (banyak gedung dilewati) justru lebih cepat daripada "sedikit hops" tapi macet.

**4. Theta ke Omicron ($\theta \to O$)**

- Jelas via Lambda ($\lambda$). $\theta \to \lambda(2) \to O(4)$ = 6 menit.
- Jika via Mu: $\theta \to \mu(5) \to O(8)$ = 13 menit.

**Simpulan Akhir:** Sistem navigasi RoboGuide sebaiknya menggunakan algoritma berbasis **Cost** (seperti A* dengan heuristik jarak nyata, atau UCS jika buta peta) daripada sekadar BFS, karena jalur terpendek secara langkah (BFS) seringkali bukan jalur tercepat secara waktu.