[T4] Keluarga Besar Pak RT

Struktur Data dan Algoritma
Departemen Teknik Komputer ITS

1 Deskripsi Masalah

Pak RT baru saja pindah ke kampung dan ingin mengenal warga di wilayahnya. Untuk memudahkan koordinasi, ia ingin membuat "pohon keluarga" warga yang menunjukkan hubungan kekeluargaan antar warga.

Dalam sistem Pak RT:

- Setiap keluarga memiliki **kepala keluarga** (root)
- Setiap orang bisa memiliki **anak** (children)
- Setiap orang (kecuali kepala keluarga) memiliki **orangtua** (parent)

Pak RT perlu bantuan untuk:

1. Menambahkan anggota keluarga baru
2. Mencari apakah seseorang adalah anggota keluarga tertentu
3. Melihat struktur pohon keluarga
4. Menghitung berapa generasi dalam keluarga

2 Format Masukan dan Keluaran

2.1 Format Masukan

Masukan terdiri atas beberapa baris. Baris pertama berisi satu buah bilangan bulat positif N ($1 \leq N \leq 20$) yang merupakan banyak operasi. N baris berikutnya masing-masing berisi operasi yang akan dilakukan dengan format sebagai berikut:

- `KELUARGA_BARU nama` - Membuat keluarga baru dengan kepala keluarga 'nama'
- `TAMBAH_ANAK parent child` - Menambahkan 'child' sebagai anak dari 'parent'
- `ADA_ORANG nama` - Mengecek apakah 'nama' ada dalam salah satu keluarga
- `CARI_ORANGTUA child` - Mencari orangtua dari 'child'
- `LIHAT_ANAK parent` - Melihat semua anak dari 'parent'
- `HITUNG_GENERASI` - Menghitung berapa generasi terdalam

Batasan:

- $1 \leq |nama| \leq 15$, hanya huruf (tanpa spasi)
- Maksimal 3 keluarga
- Maksimal 15 orang per keluarga
- Maksimal 4 generasi per keluarga

2.2 Format Keluaran

Untuk setiap operasi, keluarkan respons sesuai format berikut:

- **KELUARGA_BARU**:
  - Jika berhasil: `KELUARGA DIBUAT`
  - Jika nama sudah ada: `NAMA SUDAH ADA`
- **TAMBAH_ANAK**:
  - Jika berhasil: `ANAK DITAMBAHKAN`
  - Jika parent tidak ada: `ORANGTUA TIDAK ADA`
  - Jika child sudah ada: `NAMA SUDAH ADA`
- **ADA_ORANG**:
  - Jika ada: `ADA`
  - Jika tidak ada: `TIDAK ADA`
- **CARI_ORANGTUA**:
  - Jika ada parent: `ORANGTUA: nama_parent`
  - Jika kepala keluarga: `KEPALA KELUARGA`
  - Jika tidak ditemukan: `TIDAK DITEMUKAN`
- **LIHAT_ANAK**:
  - Jika ada anak: `ANAK: nama1 nama2 nama3`
  - Jika tidak ada anak: `TIDAK ADA ANAK`
  - Jika orang tidak ditemukan: `TIDAK DITEMUKAN`
- **HITUNG_GENERASI**: `GENERASI: angka`

3 Contoh Masukan/Keluaran

Masukan	Keluaran
10	
KELUARGA_BARU Budi	KELUARGA DIBUAT
TAMBAH_ANAK Budi Sari	ANAK DITAMBAHKAN
TAMBAH_ANAK Budi Dodi	ANAK DITAMBAHKAN
TAMBAH_ANAK Sari Andi	ANAK DITAMBAHKAN
ADA_ORANG Andi	ADA
ADA_ORANG Tono	TIDAK ADA
CARI_ORANGTUA Sari	ORANGTUA: Budi
CARI_ORANGTUA Budi	KEPALA KELUARGA
LIHAT_ANAK Budi	ANAK: Sari Dodi
HITUNG_GENERASI	GENERASI: 3

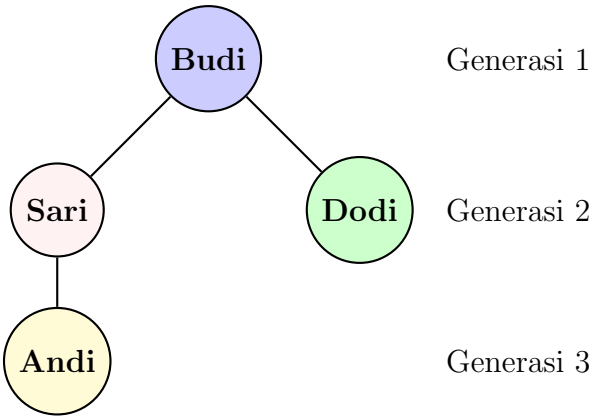
4 Penjelasan

Mari kita ikuti alur operasi pada contoh di atas:

1. **KELUARGA_BARU Budi**: Membuat keluarga baru dengan Budi sebagai kepala keluarga
2. **TAMBAH_ANAK Budi Sari**: Sari menjadi anak Budi
3. **TAMBAH_ANAK Budi Dodi**: Dodi juga menjadi anak Budi
4. **TAMBAH_ANAK Sari Andi**: Andi menjadi anak Sari (cucu Budi)
5. **ADA_ORANG Andi**: Andi ada dalam keluarga
6. **ADA_ORANG Tono**: Tono tidak ada dalam keluarga manapun
7. **CARI_ORANGTUA Sari**: Orangtua Sari adalah Budi
8. **CARI_ORANGTUA Budi**: Budi adalah kepala keluarga (tidak punya orangtua)
9. **LIHAT_ANAK Budi**: Budi punya anak Sari dan Dodi
10. **HITUNG_GENERASI**: Ada 3 generasi (Budi → Sari/Dodi → Andi)

5 Ilustrasi Pohon Keluarga

Pohon Keluarga Pak Budi



6 Strategi Implementasi

6.1 Struktur Data yang Digunakan

1. **Node Structure**: Setiap orang adalah node yang menyimpan:
   - Nama orang
   - Pointer ke parent (NULL jika kepala keluarga)
   - Array/list anak-anak
2. **Simple Tree**: Tidak perlu balanced tree, cukup pointer sederhana
3. **Linear Search**: Untuk mencari orang dalam tree

6.2 Algoritma yang Digunakan

1. **Tree Traversal**: DFS sederhana untuk mencari orang
2. **Parent-Child Relationship**: Pointer manipulation
3. **Depth Calculation**: Rekursi sederhana untuk hitung generasi

7 Contoh Struktur Node

Nama	Parent	Children
Budi	NULL	[Sari, Dodi]
Sari	Budi	[Andi]
Dodi	Budi	[]
Andi	Sari	[]

8 Algoritma HITUNG_GENERASI

Untuk menghitung generasi terdalam:

1. Mulai dari setiap kepala keluarga
2. Hitung kedalaman maksimal dengan rekursi:

- Jika node tidak punya anak, depth = 1
- Jika punya anak, depth = 1 + max(depth semua anak)

3. Return depth maksimal dari semua keluarga

9 Konsep yang Dipelajari

Soal ini mengajarkan:

1. **Tree Structure**: Node dengan parent-child relationship
2. **Pointer Manipulation**: Linking nodes together
3. **Tree Traversal**: DFS untuk search
4. **Recursion**: Untuk calculate depth
5. **Basic Tree Operations**: Insert, search, display

10 Catatan Implementasi

- Gunakan struct sederhana untuk represent node
- Tidak perlu complex tree operations seperti rotation
- Linear search cukup untuk constraints yang kecil (N15)
- Rekursi sederhana untuk depth calculation
- Validasi input: cek apakah parent exist sebelum add child
- Handle multiple families dengan array of root pointers
- Kompleksitas yang diharapkan: O(N) untuk search, O(N) untuk traversal