CHEAT SHEET DAA

Pertemuan 1 - Pseudocode

1. **Algoritma**:
    - Definisi: Serangkaian instruksi untuk mengubah input menjadi output dalam waktu terbatas.
    - Ciri-ciri: Tahapan jelas, tidak ambigu, range nilai input yang terdefinisi, dan efisiensi berbeda antara algoritma.
2. **Pseudocode**:
    - Representasi teks algoritma, menggabungkan bahasa manusia dan elemen pemrograman.
    - Mempermudah pemahaman sebelum implementasi kode.
3. **Cara Menulis Pseudocode**:
    - **Struktur Umum**: Nama algoritma, input, output, dan instruksi.
    - **Operator Penting**:
        - ← (Assignment)
        - = (Perbandingan)
        - ≠, mod, ⌈x⌉, ⌊x⌋ (Operator Matematika)
    - **Looping & Conditional**: Gunakan indentasi untuk menandai bagian dalam if-else, for, while.
4. **Tracing Algoritma**:
    - Gunakan untuk memahami variabel dan efisiensi algoritma.
    - Contoh: Algoritma Sequential Search dengan array input dan pencarian elemen tertentu.

Pertemuan 2 - Baris dan Deret

1. **Baris**:
    - Definisi: Urutan bilangan dengan aturan tertentu.
    - Rumus Suku ke-n: Menentukan nilai berdasarkan pola atau relasi perulangan (misalnya, un=4n−5u_n = 4n - 5un=4n−5).
2. **Deret**:
    - Definisi: Jumlah dari semua suku dalam barisan.
    - Rumus Jumlah Deret Aritmatika:

        - $S_n = \frac{n}{2} (a + l)$ untuk suku pertama a dan suku terakhir l.
        - Jika l tidak diketahui: $S_n = \frac{n}{2} (2a + (n-1)d)$ di mana d adalah beda.
3. **Notasi Sigma (Σ)**:
    - Penggunaan Σ untuk representasi deret terhingga, menghitung jumlah dengan mensubstitusi nilai pada variabel.
4. **Rumus Penting untuk Penjumlahan**:
    - Beberapa rumus seperti jumlah bilangan bulat $\sum i$, kuadrat $\sum i^2$, dan lainnya.
5. **Manipulasi Penjumlahan**:
    - Teknik untuk menyederhanakan deret yang kompleks agar bisa diselesaikan menggunakan rumus standar.

Pertemuan 3 - Analisis Efisiensi Algoritma Non-Rekursif

1. **Rekursif vs. Non-Rekursif**:
    - **Rekursif**: Memecah masalah menjadi kasus dasar (base case) dan kasus rekursi.
    - **Non-Rekursif (Iteratif)**: Menggunakan loop untuk menyelesaikan masalah tanpa pemanggilan fungsi secara berulang.
2. **Analisis Algoritma**:
    - Fokus pada dua aspek utama:
        - **Waktu Eksekusi**: Seberapa cepat algoritma dapat menyelesaikan tugasnya.
        - **Penggunaan Memori**: Memori yang dibutuhkan selama algoritma berjalan.
3. **Estimasi Waktu Eksekusi**:
    - Rumus umum: time = nLoop x tLoop
        - **nLoop**: Jumlah iterasi dalam loop.
        - **tLoop**: Waktu eksekusi per iterasi (dalam satuan waktu tertentu).
4. **Best, Worst, dan Average Case**:
    - **Best Case**: Kondisi paling optimal di mana algoritma selesai paling cepat (misalnya elemen ditemukan pada awal array).

- **Worst Case**: Kondisi di mana algoritma membutuhkan waktu terlama (misalnya elemen ditemukan di akhir array atau tidak ada).
- **Average Case**: Asumsi bahwa elemen dapat ditemukan di posisi mana saja dengan probabilitas yang sama.

5. **Langkah-langkah Analisis Efisiensi Waktu Algoritma Non-Rekursif**:
   - **Step 1**: Tentukan parameter input yang mempengaruhi banyaknya eksekusi (misalnya ukuran array n pada sequential search).
   - **Step 2**: Identifikasi **basic operation**, yaitu operasi yang selalu dieksekusi di setiap iterasi loop.
   - **Step 3**: Analisis apakah jumlah eksekusi basic operation bisa berbeda untuk ukuran input yang sama (analisis kasus terbaik, terburuk, dan rata-rata).
   - **Step 4**: Tentukan rumus deret untuk jumlah eksekusi basic operation (misalnya C(n) = n untuk worst case pada sequential search).
   - **Step 5**: Selesaikan rumus untuk menghitung jumlah eksekusi basic operation di setiap kasus.
6. **Rumus Eksekusi untuk Sequential Search**:
   - **Best Case**: C(n)=1C(n) = 1C(n)=1 (elemen ditemukan di awal).
   - **Worst Case**: C(n)=nC(n) = nC(n)=n (elemen ditemukan di akhir atau tidak ada).
   - **Average Case**: C(n)=n+12C(n) = \frac{n + 1}{2}C(n)=2n+1.
7. **Estimasi Waktu Running Total**:
   - Formula: T(n)=Cop×C(n)T(n) = Cop \times C(n)T(n)=Cop×C(n)
     - **T(n)**: Total waktu eksekusi untuk input berukuran n.
     - **Cop**: Waktu eksekusi untuk satu basic operation (biasanya dianggap 1 satuan waktu).

Pertemuan 4 - Algoritma Rekursif

1. **Algoritma Rekursif:**
   - Fungsi yang memanggil dirinya sendiri di dalam definisinya.
   - Rekursi Langsung: Fungsi memanggil dirinya sendiri secara langsung.
   - Rekursi Tidak Langsung: Fungsi memanggil fungsi lain yang kemudian memanggil fungsi pertama kembali.

2. **Ciri-Ciri Masalah Rekursif:**
    - Base Case (Kasus Dasar): Kasus sederhana yang bisa diselesaikan tanpa rekursi (misalnya, n!=1n! = 1n!=1 untuk n=1n = 1n=1).
    - Recursive Case (Kasus Rekursif): Memecah masalah menjadi kasus yang lebih kecil, bergerak menuju kasus dasar (misalnya, n!=n×(n−1)!n! = n \times (n-1)!n!=n×(n−1)!).
3. **Keuntungan Algoritma Rekursif:**
    - Mempermudah kode untuk masalah yang kompleks.
    - Seringkali lebih mudah dibaca dibanding versi iteratif.
4. **Kekurangan:**
    - Penggunaan memori lebih besar karena memerlukan stack frame.
    - Bisa lebih sulit dilacak dan di-debug.
5. **Contoh:**
    - Faktorial:
        - Fungsi rekursif: jika n <= 1 maka return 1, jika tidak return n * faktorial(n - 1).
    - Perkalian dengan Penjumlahan:
        - Fungsi rekursif: a * b = (a - 1) * b + b.

Pertemuan 5 - Relasi Perulangan

1. **Relasi Perulangan (Recurrence Relation):**
    - Sebuah formula yang mendefinisikan setiap suku dalam barisan berdasarkan suku-suku sebelumnya.
    - Berguna untuk mendefinisikan barisan yang mengikuti pola tertentu.
2. **Contoh Relasi Perulangan:**
    - Barisan Aritmatika: an=an−1+da_n = a_{n-1} + dan=an−1+d
    - Barisan Geometri: an=r×an−1a_n = r \times a_{n-1}an=r×an−1
3. **Metode Penyelesaian Relasi Perulangan:**
    - Substitusi Maju (Forward Substitution): Mengembangkan suku-suku secara maju dari suku awal.
    - Substitusi Mundur (Backward Substitution): Menghitung mundur dari suku yang diinginkan.

4. **Contoh Perhitungan:**
    - Contoh Barisan Aritmatika: Barisan: 1, 3, 5, 7,... dengan an=an−1+2a_n = a_{n-1} + 2an=an−1+2, suku awal a1=1a_1 = 1a1=1.
    - Contoh Barisan Geometri: Barisan: 3, 9, 27,... dengan an=3×an−1a_n = 3 \times a_{n-1}an=3×an−1, suku awal a1=3a_1 = 3a1=3.

Pertemuan 5 - Analisa Efisiensi Algoritma Rekursif

1. **Contoh Algoritma Rekursif:**
    - Algoritma Fibonacci: Menghitung bilangan Fibonacci ke-n secara rekursif.
    - Algoritma Pangkat: Menghitung XnX^nXn menggunakan rekursi.
2. **Langkah-Langkah Analisis Efisiensi Algoritma Rekursif:**
    - Tentukan Metrik Ukuran Input: Faktor yang menentukan kompleksitas, misalnya nnn pada fungsi rekursif.
    - Identifikasi Basic Operation: Operasi dasar yang dilakukan setiap kali fungsi rekursif dipanggil.
    - Analisa Eksekusi Basic Operation: Tentukan apakah jumlah eksekusi operasi dasar akan selalu sama atau bervariasi.
    - Persamaan Rekursi Basic Operation: Bentuk persamaan rekursi yang menunjukkan frekuensi eksekusi operasi dasar.
    - Cari Rumus Langsung Eksekusi Basic Operation: Menemukan hubungan langsung yang menyatakan jumlah operasi.
3. **Contoh Analisis pada Algoritma Pangkat (Recursive Power):**
    - Basic Operation: Operasi perkalian X * pangkat(X, n-1).
    - Persamaan Rekursif untuk Basic Operation:
        - Jika C(n)C(n)C(n) adalah banyaknya operasi dasar untuk input nnn:
            - C(n)=C(n−1)+1C(n) = C(n-1) + 1C(n)=C(n−1)+1 untuk n>1n > 1n>1.
            - C(1)=1C(1) = 1C(1)=1 (base case).
    - Menghitung Jumlah Operasi:
        - Rumus Langsung: Dari rekursi tersebut, didapatkan:
            - C(2)=2C(2) = 2C(2)=2, C(3)=3C(3) = 3C(3)=3, C(4)=4C(4) = 4C(4)=4, dan seterusnya.
        - Artinya, jumlah operasi C(n)=nC(n) = nC(n)=n.

4. **Pohon Rekursif:**
    - Pohon rekursif menunjukkan jumlah panggilan dan operasi yang dilakukan pada setiap tingkat rekursi.
    - Contoh pada algoritma pangkat 545^454, dimana setiap tingkat menunjukkan pengulangan hingga mencapai kondisi dasar.