**Contoh desain dan analisis**

**1. Teknik Recursif:**

**Contoh Desain**: Fungsi rekursif untuk menghitung jumlah angka dalam sebuah array.

**Analisis**: Setiap langkah mengurangi ukuran masalah dan memanggil dirinya sendiri hingga mencapai kondisi dasar. Kompleksitas waktu: O(n), kompleksitas ruang: O(n) (karena memanggil fungsi rekursif).

**2.Teknik Backtracking:**

**Contoh Desain**: Backtracking untuk menyelesaikan puzzle Sudoku.

**Analisis**: Setiap langkah mencoba nilai yang mungkin dan memeriksa apakah solusi masih memungkinkan. Efisiensi bergantung pada pemilihan nilai dan pemotongan cabang. Kompleksitas waktu dan ruang bervariasi tergantung pada implementasi.

**3. Metode Divide and Conquer:**

**Contoh Desain**: Algoritma QuickSort untuk pengurutan array.

**Analisis**: Memecah array menjadi dua bagian, menyortir masing-masing bagian, dan menggabungkan hasilnya. Kompleksitas waktu: O(n log n), kompleksitas ruang: O(log n) (karena memanggil fungsi rekursif).

**4. Metode Greedy:**

**Contoh Desain**: Algoritma Greedy untuk penyelesaian masalah Ransum dengan memilih makanan dengan nilai gizi tertinggi per berat.

**Analisis**: Pada setiap langkah, memilih opsi terbaik secara lokal. Kompleksitas waktu: O(n log n), kompleksitas ruang: O(1) (bergantung pada implementasi).

**5. Metode Binary Search Tree:**

**Contoh Desain**: Binary Search Tree untuk penyisipan dan pencarian elemen.

**Analisis**: Pencarian, penyisipan, dan penghapusan dapat memiliki kompleksitas waktu O(log n) pada rata-rata, tetapi O(n) dalam kasus terburuk jika pohon tidak seimbang. Kompleksitas ruang: O(n).

**6. Metode Branch and Bound:**

**Contoh Desain**: Branch and Bound untuk masalah Knapsack.

**Analisis**: Mengevaluasi batas-batas dan memotong cabang yang tidak menjanjikan. Efisiensi tergantung pada pemilihan batas-batas. Kompleksitas waktu dan ruang tergantung pada implementasi.

**7.Teknik Searching secara Paralel:**

**Contoh Desain**: Pencarian linear di beberapa thread.

**Analisis**: Pencarian dilakukan secara bersamaan di beberapa bagian untuk meningkatkan kecepatan. Analisis efisiensi dan keseimbangan beban.

**8. Teknik Sorting secara Paralel:**

**Contoh Desain**: Parallel Merge Sort.

**Analisis**: Proses pengurutan dilakukan secara bersamaan di beberapa bagian data. Analisis efisiensi dan kecepatan tergantung pada bagaimana pekerjaan dibagi dan disatukan secara paralel.