Penjelasan Desain dan Analisis

1.Teknik Recursif:

Desain: Algoritma rekursif sering mengikuti struktur berulang dari masalah, memecahnya menjadi submasalah yang lebih kecil hingga mencapai kasus dasar yang dapat dipecahkan langsung.

Analisis: Penting untuk memastikan bahwa pemanggilan rekursif memiliki basis yang solid dan mencapai kasus dasar dengan jumlah langkah yang terbatas. Analisis kompleksitas waktu dan ruang diperlukan untuk memahami kinerja algoritma.

2.Teknik Backtracking:

Desain: Backtracking berfokus pada pencarian solusi melalui percobaan dan pengulangan, dengan kemungkinan "mengembalikan" langkah-langkah yang tidak mengarah ke solusi.

Analisis: Evaluasi efisiensi backtracking melibatkan identifikasi langkah-langkah yang dapat "dipotong" untuk menghindari eksplorasi yang tidak perlu dan pemahaman tentang kondisi berhenti untuk mencapai solusi.

3.Metode Divide and Conquer:

Desain: Algoritma ini memecah masalah menjadi submasalah yang lebih kecil, menyelesaikannya secara rekursif, dan menggabungkan solusi untuk memperoleh solusi keseluruhan.

Analisis:Penting untuk mengevaluasi efisiensi pembagian masalah dan penggabungan solusi. Analisis kompleksitas waktu dan ruang membantu memahami kinerja algoritma.

4. Metode Greedy:

Desain:Greedy algorithms membuat keputusan lokal yang diharapkan memberikan solusi yang optimal secara global. Setelah suatu keputusan diambil, tidak ada revisi.

Analisis: Memerlukan bukti bahwa memilih opsi terbaik pada setiap langkah menghasilkan solusi global yang optimal. Analisis ketidakambangan penting untuk memastikan keandalan algoritma.

5.Metode Umum:

Desain: Struktur data yang memungkinkan pencarian, penyisipan, dan penghapusan dalam waktu logaritmik. Setiap simpul memiliki dua anak: satu di sebelah kiri (nilai lebih kecil) dan satu di sebelah kanan (nilai lebih besar).

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

6.Metode Branch and Bound:

Desain: Metode ini menggunakan strategi pemotongan cabang yang tidak menjanjikan untuk mengurangi ruang pencarian. Sering digunakan dalam masalah optimasi.

Analisis:Evaluasi pemilihan cabang dan teknik peringkat bound adalah kunci untuk memahami kinerja dan optimalitas algoritma. Diperlukan pemahaman tentang cara membatasi pencarian.

7.Teknik Searching secara Paralel:

Desain: Pencarian dilakukan secara bersamaan di beberapa bagian ruang pencarian untuk meningkatkan kecepatan pencarian.

Analisis: Memerlukan analisis keseimbangan beban, manajemen konflik, dan pembagian pekerjaan untuk memastikan bahwa pencarian dapat dilakukan secara efisien dan akurat.

8.Teknik Sorting secara Paralel:

Desain: Proses pengurutan dilakukan secara bersamaan di beberapa bagian data untuk meningkatkan kecepatan pengurutan.

Analisis: Penting untuk memastikan bahwa hasil pengurutan dari setiap bagian dapat digabungkan dengan efisien dan akurat. Analisis kompleksitas waktu dan ruang membantu memahami efisiensi algoritma pengurutan.