NAMA : Restu Wibisono
NPM : 2340506061

ALGORITMA GREEDY

I. PENGERTIAN

Algoritma greedy adalah strategi pemecahan masalah yang memilih opsi terbaik pada setiap langkahnya dengan harapan mencapai solusi keseluruhan yang optimal. Meskipun sederhana dan efisien, algoritma ini tidak menjamin optimalitas solusi untuk setiap permasalahan yang dihadapi.

1. Scheduling:

   Dalam masalah scheduling, tujuannya adalah mengalokasikan sumber daya atau tugas ke slot waktu yang tersedia. Algoritma greedy dapat digunakan untuk memprioritaskan tugas berdasarkan kriteria tertentu, seperti waktu mulai tercepat atau durasi terpendek. Algoritma ini memilih tugas dengan prioritas tertinggi dan menempatkannya pada slot waktu yang tersedia, berlanjut hingga semua tugas terjadwal.

2. Knapsack:

   Masalah knapsack melibatkan pemilihan subset item dengan nilai maksimum, dengan kapasitas terbatas. Algoritma greedy untuk masalah knapsack membuat keputusan berdasarkan rasio nilai terhadap berat setiap item. Pendekatan ini memilih item dengan rasio nilai-berat tertinggi hingga kapasitas knapsack terpenuhi. Meskipun tidak selalu menghasilkan solusi optimal, algoritma ini dapat efisien untuk beberapa kasus.

3. Set Covering:

   Set covering adalah masalah di mana sekelompok elemen harus ditutupi dengan memilih subset minimum. Algoritma greedy untuk set covering dimulai dengan solusikosong dan secara iteratif menambahkan subset yang mencakup elemen paling banyak yang belum tercakup. Proses ini berlanjut hingga semua elemen tercakup atau tidak ada subset lagi yang dapat ditambahkan. Meskipun tidak selalu optimal, algoritma ini dapat memberikan perkiraan solusi yang cukup baik.

II. LANGKAH-LANGKAH

1. Scheduling (Penjadwalan):
   - Langkah 1: Urutkan tugas berdasarkan waktu mulai atau deadline.

- Langkah 2: Pilih tugas dengan waktu mulai terkecil atau deadline terdekat.
- Langkah 3: Jika ada tumpang tindih dengan tugas yang sudah dipilih, pilih tugas dengan waktu selesai terkecil.
- Langkah 4: Ulangi langkah 2 dan 3 sampai semua tugas terjadwal.

2. Knapsack (Ransel):
    - Langkah 1: Hitung rasio nilai-berat (value/weight) untuk setiap item.
    - Langkah 2: Urutkan item berdasarkan rasio nilai-berat secara menurun.
    - Langkah 3: Pilih item dengan rasio nilai-berat tertinggi dan masukkan ke dalam ransel jika masih ada kapasitas tersisa.
    - Langkah 4: Ulangi langkah 3 sampai ransel penuh atau tidak ada item lagi.
3. Set Covering (Penutup Himpunan):
    - Langkah 1: Buat himpunan kosong untuk menampung elemen yang belum ditutup.
    - Langkah 2: Pilih himpunan yang mencakup sebagian besar elemen yang belum ditutup.
    - Langkah 3: Tambahkan himpunan tersebut ke dalam solusi dan hapus elemen yang sudah ditutup.
    - Langkah 4: Ulangi langkah 2 dan 3 sampai semua elemen tercakup.

III. ISTILAH-ISTILAH

1. Greedy Choice Property: Karakteristik algoritma Greedy yang memilih langkah terbaik pada setiap langkahnya tanpa mempertimbangkan dampak jangka panjang.
2. Optimal Substructure: Sifat masalah yang dapat dipecahkan dengan membaginya menjadi submasalah yang lebih kecil dan menyelesaikan masing-masing submasalah secara optimal.
3. Subproblem: Permasalahan yang lebih kecil yang harus diselesaikan sebagai bagian dari solusi masalah yang lebih besar.
4. Solution Set: Kumpulan solusi yang dihasilkan oleh algoritma Greedy.
5. Feasible Solution: Solusi yang memenuhi semua batasan dan persyaratan masalah.
6. Objective Function: Fungsi matematis yang digunakan untuk mengukur kualitas solusi.
7. Local Optimum: Solusi terbaik yang dapat dicapai pada setiap langkah, namun tidak selalu menghasilkan solusi global terbaik.
8. Global Optimum: Solusi terbaik yang dapat dicapai untuk suatu masalah tertentu.

9. Non-greedy Algorithm: Algoritma yang tidak menggunakan sifat serakah dan mempertimbangkan dampak jangka panjang pada setiap langkahnya.
10. Heuristic Algorithm: Algoritma yang menghasilkan solusi yang cukup baik, tetapi tidak selalu optimal, dengan menggunakan aturan praktis atau pengalaman sebelumnya.

IV. MASALAH YANG DAPAT DISELESAIKAN

1. Masalah Jalan Terpendek (Shortest Path Problem): Algoritma Greedy dapat diterapkan untuk menemukan jalan terpendek antara dua titik dalam grafik. Pada setiap tahap, algoritma Greedy akan memilih simpul terdekat yang belum dikunjungi sebagai langkah berikutnya.
2. Masalah Jadwal (Scheduling Problem): Algoritma Greedy dapat digunakan untuk menyusun jadwal kegiatan dengan memprioritaskan kegiatan yang memiliki waktu selesai paling awal atau durasi paling pendek terlebih dahulu.
3. Masalah Pemotongan Batang (Cutting Stock Problem): Algoritma Greedy bisa digunakan untuk memotong batang dengan panjang tertentu menjadi beberapa potongan yang diinginkan, dengan memilih potongan terpanjang yang memungkinkan pada setiap tahap.
4. Masalah Pemilihan Aktivitas (Activity Selection Problem): Algoritma Greedy bisa diterapkan untuk memilih subset aktivitas yang saling kompatibel dengan memprioritaskan aktivitas dengan waktu selesai paling awal pada setiap tahap.
5. Masalah Pemilihan Koin (Coin Change Problem): Algoritma Greedy dapat digunakan untuk mencari jumlah koin minimum yang diperlukan untuk memberikan kembalian tertentu, dengan memilih koin berdenominasi terbesar yang memungkinkan pada setiap tahap.

V. KELEBIHAN

1. Sederhana dan Mudah Dipahami: Algoritma Greedy umumnya memiliki struktur yang simpel dan mudah dipahami. Konsep intinya adalah memilih langkah terbaik pada setiap iterasi tanpa mempertimbangkan implikasi jangka panjang. Hal ini membuat implementasi dan pemahaman algoritma Greedy lebih mudah bagi para pemrogram.
2. Efisiensi Waktu: Algoritma Greedy sering kali menunjukkan kompleksitas waktu yang lebih rendah dibandingkan dengan algoritma yang lebih kompleks. Karena hanya mempertimbangkan langkah terbaik pada setiap langkah, algoritma Greedy tidak

memerlukan perhitungan rumit atau pencarian solusi secara eksploratif, sehingga efisien dalam eksekusi waktu.

3. Solusi Cukup Baik: Meskipun tidak selalu menghasilkan solusi optimal, algoritma Greedy cenderung memberikan solusi yang memadai dalam banyak kasus. Algoritma ini dapat mendekati solusi optimal atau memberikan solusi yang memenuhi kebutuhan praktis.
4. Dapat Digunakan untuk Masalah Optimisasi: Algoritma Greedy sangat sesuai untuk masalah optimisasi di mana tujuannya adalah mencari solusi terbaik pada setiap langkah. Algoritma ini dapat digunakan untuk mencari solusi terpendek, terpanjang, terkecil, terbesar, atau solusi yang memaksimalkan atau meminimalkan suatu fungsi objektif.
5. Fleksibilitas: Algoritma Greedy dapat dengan mudah dimodifikasi atau dikombinasikan dengan algoritma lain untuk menyelesaikan masalah yang lebih kompleks. Algoritma Greedy dapat berfungsi sebagai langkah awal dalam algoritma yang lebih kompleks atau sebagai komponen dari pendekatan kombinasi untuk mencapai solusi yang lebih baik.

VI. KEKURANGAN

1. Tidak Selalu Optimal: Algoritma Greedy tidak selalu menghasilkan solusi optimal untuk setiap masalah. Karena algoritma ini hanya mempertimbangkan langkah terbaik pada setiap langkahnya tanpa memperhitungkan konsekuensi jangka panjang, hasilnya mungkin tidak memenuhi persyaratan masalah atau tidak mencapai solusi global terbaik.
2. Tidak Mempertimbangkan Alternatif: Algoritma Greedy cenderung memilih langkah terbaik pada setiap iterasi tanpa mempertimbangkan alternatif yang mungkin lebih baik di langkah berikutnya. Ini dapat mengakibatkan algoritma Greedy terjebak pada solusi lokal yang suboptimal dan gagal mencapai solusi global terbaik.
3. Memerlukan Greedy Choice Property: Algoritma Greedy hanya dapat diterapkan jika masalah memenuhi sifat Greedy Choice Property, yaitu memilih langkah terbaik pada setiap langkahnya akan menghasilkan solusi optimal secara keseluruhan. Jika masalah tidak memenuhi sifat ini, algoritma Greedy tidak dapat digunakan atau menghasilkan solusi yang tidak memenuhi persyaratan masalah.

4. Tidak Dapat Mengembalikan Keputusan: Algoritma Greedy tidak dapat membatalkan keputusan yang telah diambil pada langkah sebelumnya. Setelah langkah diambil, keputusan tersebut tidak dapat diubah atau diperbaiki. Hal ini dapat menyebabkan algoritma Greedy terjebak pada solusi yang tidak optimal atau tidak memenuhi persyaratan masalah.
5. Tidak Toleran terhadap Perubahan: Algoritma Greedy kurang fleksibel terhadap perubahan dalam masalah atau kondisi. Jika terjadi perubahan dalam masalah atau persyaratan, algoritma Greedy mungkin tidak mampu menghasilkan solusi yang sesuai dengan perubahan tersebut. Dalam beberapa situasi, modifikasi atau penggunaan algoritma lain yang lebih sesuai dapat diperlukan untuk menangani perubahan tersebut.