*19 / 09 / 2023*

*Day – 16*

**Selection Sort, Array & Linked List**

* **Bagaimana Memory Bekerja**
* Semisal saja disini kita menghadiri suatu pertunjukan dalam konteks ini adalah pertunjukan teater dan barang bawaan kita harus dititipkan dan dismpan ke dalam suatu lemari yang berisi sekumpulan laci.
* Setiap laci dapat menyimpan 1 item atau 1 barang atau 1 elemen.
* Kita ingin menyimpan 2 barang, jadi kita meminta 2 laci.
* Kita menyimpan 2 barang kita dilaci yang kita pesan. Barang pertama yang disimpan adalah payung dan barang kedua adalah boneka kelinci.
* Ternyata ini serupa dengan bagaimana cara memori pada computer kita bekerja.
* Memori pada computer kita itu bisa digambarkan sebagai sebuah lemari raksasa yang terdiri dari banyak laci dimana setiap laci nya akan memiliki alamat (address).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address  001 | Address  002 | Address  fe0ffeeb | Address  004 | Address  005 |
| Address  006 | Address  007 | Address  008 | Address  009 | Address  010 |
| Address  011 | Address  012 | Address  013 | Address  014 | Address  015 |
| Address  016 | Address  017 | Address  018 | Address  019 | Address  020 |
| Address  021 | Address  022 | Address  023 | Address  024 | Address  025 |

* Setiap kali kita butuh untuk menyimpan sesuatu dimemori, kita meminta pada komputer untuk menyediakan satu space pada suatu ruangan dan komputer akan memberikan suatu alamat ke kita untuk menyimpan item kita atau dalam konteks ini berupa data.
* **Struktur Data Array**
* Adakala nya kita butuh untuk menyimpan sekumpulan nilai ke dalam memori.
* Todo app ini biasanya digunakan untuk mencatat beberapa point penting yang perlu kita lakukan atau perlu kita kerjakan.
* Disini berarti untuk membangun sebuah Todo app semacam ini kita butuh untuk menyimpan sekumpulan nilai didalam memori.
* Untuk kasus kita disini, kita akan menyimpan data dari todos kita ke dalam suatu array.
* Menggunakan array berarti kita akan menyimpan setiap data dari todos app ini secara berdampingan satu dengan yang lain nya di dalam memori.
* Disini kata kunci nya adalah **data dalam suatu array akan disimpan secara tepat berdampingan satu dengan yang lain nya di dalam memori.**
* Semisal todo app kita akan menyimpan 3 buah data yaitu brunch, bocce dan tea.
* Kalau ketiga data ini disimpan ke dalam array maka data ini akan tersimpan di memori secara tepat bersebelahan satu dengan yang lain nya.
* Kita butuh untuk menambahkan data keempat pada todolist kita. Tetapi jika kita melihat isi memori nya area berikut nya sudah di alokasi kan untuk data yang lain.
* Untuk kasus array jika kita ingin menambahkan data keempat, data keempatnya tidak bisa langsung ditempatkan disini.
* Kenapa ? karena array mensyaratkan agar data yang disimpan ke dalam memori itu harus tepat bersebelahan satu dengan yang lain nya.
* Artinya jika kita menghadapi kondisi semacam ini kita perlu mencari area baru di memori yang memang belum teralokasi dan cukup untuk menampung 4 buah data.
* Selanjutnya ketika data yang lama akan disalin ke area yang baru beserta data keempat.
* Menambahkan data baru ke dalam suatu array bisa jadi cukup bermasalah.
* Ketika kita ingin menambahkan data baru ke dalam array dan memang sudah kehabisan space artinya kita butuh untuk mencari spot baru atau lokasi baru dimemori sehingga penambahan data baru ini menjadi sangat lambat.
* Solusi yang cukup mudah untuk mengatasi permasalahan ini adalah dengan melakukan reservasi alokasi memori atau istilah nya adalah “hold seats”.
* Artinya walaupun kita hanya memiliki 3 data yang perlu disertakan ke dalam todolist kita, kita akan memita komputer untuk mengalokasikan misal saja 10 slot atau 10 area, ini dimaksudkan untuk jaga – jaga . Dengan demikian kita bisa menambahkan sampai dengan 10 data ke dalam todolist kita tanpa harus berpindah ke alokasi memori yang lain.
* Disini kita dihadapkan pada 2 kelemahan dri pendekatan reservasi ini:
* Yang pertama adalah andai kata kita tidak menggunakan extra slot yang tersedia, maka alokasi memori tersebut menjadi sia – sia atau terbuang. Dimana kita tidak menggunakan alokasi memori tersebut tetapi program lain atau aplikasi lain juga tidak dapat menggunakannya.
* Lalu pada point yang kedua, bisa jadi suatu saat todolist kita butuh menampung lebih dari 10 data atau 10 item artinya mau tidak mau kita juga harus kita juga harus mencari alokasi memori yang baru.
* Ini adalah konsep dasar dari array dan cara kerja nya.
* **Struktur Data Linked List**
* Pada Linked List data kita oleh ditempatkan dimanapun juga di memori kita.
* Nah sebagai contoh, disini yah ini alokasi memori kita lalu data pertama kita tempatkan disini (data brunch) lalu data bocce nya ditempatkan disisi yang lain demikian juga data tea nya yang juga ditempatkan secara acak di memori atau dengan kata lain ketiga data ini tidak harus ditempatkan bersebelahan satu dengan yang lainnya.
* Ini juga perbedaan mendasar antara array dengan linked list.
* Walaupun setiap data boleh tersimpan dimanapun juga pada memori kita tetapi ada satu hal yang perlu diperhatikan dimana setiap item nya atau setiap data nya itu juga menyimpan alamat dari data berikutnya.
* Sebagai contoh disini data brunch akan menyimpan informasi mengenai alamat dari data bocce demikian juga dengan data bocce akan menyimpan informasi terkait alamat dari data tea demikian seterusnya.
* Menambahkan data ke dalam suatu linked list itu bisa terbilang sangat mudah yang perlu kita lakukan adalah kita bisa menempatkan data kita dimanapun juga dalam memori tentu nya selama alokasi memori nya maish tersedia.
* Informasi alamat dari data baru tersebut harus disertakan pada data sebelumnya.
* Dengan demikian pada dasarnya sewaktu kita menggunakan linked list kita sama sekali tidak perlu memindahkan posisi data dari data kita.
* **Struktur Data Array Vs Struktur Data Linked List**
* Disini kita dihadapkan dengan suatu kebutuhan untuk membaca item terakhir atau data terakhir darti linked list.
* Kita tidak bisa langsung mengakses data paling akhir yang tersimpan pada linked list kita. Kita harus melakukan proses pembacaan dari data pertama lalu dari data pertama kita akan mengetahui alamat dari data kedua dari data kedua kita baru mengetahui alamat dari data ketiga lalu sewaktu kita mengunjungi data ketiga kita baru mengetahui alamat data keempat demikian seterusnya sampai kita bisa mengakses data paling akhir.
* Linked List merupakan struktur data yang tepat bila kita dihadapkan pada suatu kebutuhan untuk membaca keseluruhan data dalam sewaktu – waktu. Karena disini kita akan membaca mulai dari item pertama, lalu berlanjut ke item kedua, ketiga dan seterusnya sampai ke item paling akhir.
* Tetapi jika kita butuh untuk mengakses data secara acak karena kita mengakses data pertama, karena kita ingin mengakses data ketiga, karena kita ingin mengakses data terakhir maka linked list bukanlah pilihan terbaik bahkan bisa dibilang linked list merupakan pilihan yang buruk.
* Pada array, kita akan mengetahui alamat dari setiap item yang tersimpan pada array kita.
* Semisal saja kita memiliki sebuah array yang berisi 5 item dan kita tahu bahwa array tersebut dimulai dari alamat 00.
* Artinya kalo kita butuh mengakses item kelima maka kita bisa dengan mudah mengetahui bahwa alamat dari item kelima adalah 04.
* Array itu merupakan struktur data yang tepat bila kita butuh untuk melakukan random access atau membaca elemen secara acak karena kebutuhan nya adalah hanya untuk membaca elemen pada posisi tertentu tanpa berurutan, dan dengan menggunakan array maka kita bisa melakukan hal ini dengan sangat mudah sekali.
* Berbeda kasus nya dengan linked list, karena pada linked list setiap elemen itu tidak disimpan bersebelahan satu dengan yang lainnya pada memori, kita tidak bisa dengan mudah mencari tahu alamat dari posisi data tertentu.
* **Penambahan Data Di Tengah (Insert)**
* Kita dihadapkan dengan kebutuhan yang lain yaitu menambahkan data baru ke dalam suatu list tetapi posisi nya bukan diakhir melainkan ditengah atau diantara data yang sudah tersimpan didalam todolist kita.
* Manakah yang lebih baik atau manakah yang lebih mudah untuk menambahkan suatu elemen di bagian tengah dengan menggunakan array atau menggunakan linked list ?
* Dan jawaban nya adalah list, jika kita menggunakan list atau linked list proses nya menjadi sangat mudah sekali karena yang perlu kita lakukan adalah kita hanya perlu melakukan modifikasi terhadap pointer dari elemen sebelumnya.
* Jika kita menggunakan array ini lumayan repot Ketika kita ingin menambahkan data baru diantara data yang sudah ada, kita perlu melakukan memidahkan data yang sudah ada ke setelah nya.
* Ketika space yang dibutuhkan tidak ada yang kosong, kita perlu mengcopy semua alokasi space yang baru.
* Pada kasus ini linked list lebih tepat ketimbang array karena ketika kita ingin menambahkan sebuah elemen atau data tersebut bisa disimpan ke tengah.
* Disini kita bandingkan Big O atau run time untuk array maupun list.

|  |  |  |
| --- | --- | --- |
| **PERFORMANCE** | **ARRAY** | **LISTS** |
| **Reading** | O(1) | O(n) |
| **Insertion** | O(n) | O(1) |

Catatan:

O(n) = Linear Time

O(1) = Constant Time

* Constant Time adalah jumlah waktu yang dibutuhkan itu sama sekali tidak dipengaruhi oleh jumlah data nya.
* **Deletion**
* Jika kita melakukan operasi deletion atau penghapusan elemen mana yang lebih baik list atau kah array ?
* List lebih baik lebih baik bila dibandingkan array, karena suatu kita menghapus suatu elemen pada suatu list yang perlu kita lakukan adalah kita hanya butuh untuk mengubah pointer dari elemen sebelumnya.
* Sedangkan pada array kita harus memindahkan semua elemen – elemen berikutnya ke posisi yang lebih depan atau istilah nya adalah move up.
* Berbeda kasus nya dengan insertion atau penambahan data baru, proses penghapusan data bisa dipastikan selalu berhasil.
* Karena proses penambahan data baru ada kemungkinan gagal nah kondisi gagal nya ini itu salah satu nya bisa disebabkan karena ketika tidak ada lagi space yang tersisa dari memori kita atau dengan kata lain semua space dari memori nya sudah teralokasikan tetapi berbeda hal nya dengan pengahpusan data proses penghapusan data bisa dipastikan akan selalu berhasil karena disini kita tidak membutuhkan alokasi memori yang baru melainkan disini justru kita membebaskan suatu alokasi memori.
* Disini kita tambahkan pada table kita yah perbedaan performa untuk proses penghapusan data baik pada array maupun linked list.

|  |  |  |
| --- | --- | --- |
| **PERFORMANCE** | **ARRAY** | **LISTS** |
| **Reading** | O(1) | O(n) |
| **Insertion** | O(n) | O(1) |
| **Deletion** | O(n) | O(1) |

* Proses insertion dan deletion akan memiliki performa constant time O(1) dengan catatan kita melakukan akses secara langsung terhadap elemen yang akan kita hapus tadi.
* Suatu hal yang cukup umum ditemui pada suatu linked list untuk memantau posisi pertama dan posisi terakhir nya ini terkait dengan deletion atau penghapusan data.

* **Dua Tipe Akses Sequential Access & Random Access**
* Terdapat 2 model atau 2 tipe pengaksesan data yang pertama random access dan yang kedua adalah sequential access.
* Sequential Access adalah proses pembacaan elemen satu per satu dimulai dari elemen pertama.
* Linked List itu hanya bisa dilakukan sequential access dimana Ketika kita butuh membaca elemen ke-10 maka kita terlebih dahulu harus membaca kesembilan elemen sebelumnya, dan pada elemen ke-9 baru kita memperoleh informasi terkait posisi dari elemen kesepuluh nya.
* Random Access adalah kita bisa membaca suatu elemen tertentu secara langsung tanpa harus mengunjungi elemen – elemen sebelumnya.
* Array itu lebih cepat dalam proses pembacaan data. Ini dikarenakan model akses data yang diusung suatu array adalah random access