**REVIEW MATERI**

1. Struktur data STL C++

* vector <>

vector merupakan array dinamis , yang mana size nya bisa berkurang atau bertambah sesuai kebutuhan, untuk mengakses nya sama seperti array biasa dengan syarat index nya sudah pernah di push\_back() , operasi umum :

* + v.push\_back(x) 🡪 masukkan x ke index terkahir vector v
  + v.pop\_back() 🡪 pop/ keluarkan index terakhir vector v
  + v[x] 🡪 akses vector v ke-x dengan syarat index ke-x sudah pernah di push\_back()

vector berguna untuk permasalahan yang membutuhkan input data yang dinamis, contoh penting yaitu untuk construct graph (Adjacency list).

* Stack & Queue

Stack(tumpukan) dan Queue(Antrian) merupakan dua struktur data yang cukup penting , operasinya yaitu :

* + Stack :
    - st.push(x) -> masukan x ke stack st
    - st.pop()-> keluarkan elemen teratas dari stack st
    - st.top() -> akses stack st (elemen teratas)
  + Queue:
    - st.push(x) -> masukan x ke stack st
    - st.pop()-> keluarkan elemen teratas dari stack st
    - st.front() -> akses stack st (elemen terdepan)

untuk menghapus isi stack/queue bisa dnegan perintah st.clear(), stack dan queue berguna di banyak hal , contoh pasti nya yaitu dfs(stack) dan bfs(queue)

* map
  + map merupakan struktur data yang mana untuk mengaksesnya jauh lebih fleksibel , kita dapat menggunakan char,string maupun integer besar atau negative sebagai index
    - akses : a=map[“saya”]
* Set
  + Himpunan, yaitu struktur data yang berfungsi utama untuk mencek apa suatu item terdapat dalam himpunan?
    - Untuk memasukkan data yaitu v.push(x) , sedangkan untuk mencek yaitu v.find()
* Struct
  + Struct merupakan tipe data bentukan, didalamnya dapat kita kombinasikan dengan tipe data dan struktur data yang tersedia.
  + Contoh :
    - Struct tipe{

int x,y,z;

string s;

vector <int> g;

}

tipe a //variable a bertipe data ‘tipe’

a.x=a.y.=a.z=0 //ketiganya bertipe integer

a.s=”saya”//string

a.g.push\_back(1) // vector didalam variable

untuk lebih jelas , silahkan buka cplusplus.com dan bereksperimen sendiri ☺

soal related dengan materi diatas :

<https://pandaoj.com/problem/EMAIL>

<https://pandaoj.com/problem/QUEUE>

1. Sieve of Erathostenes

Sieve of Erathostenes merupakan algoritma untuk generate bilangan prima dari [2,N]. simpelnya , algoritma ini menghapus seluruh kelipatan dari bilangan-bilangan [2,N] , sisanya yang tidak mengalami proses penghapusan merupakan bilangan prima. Untuk lebih lengkapnya klik <https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes>

Interesting usage :

* Sieve pada intinya digunakan untuk mengenerate bilangan prima, tetapi kita dapat memodifikasi algoritma tersebut menjadi mempunyai kegunaan lain seperti mendapatkan jumlah dari factor-faktor suatu bilangan(contoh : 10 , faktornya yaitu 10,5,2,1 jumlahnya 10+5+2+1=18) , proses pengeditan ada dalam bagian f[j]=true (diganti dengan f[j]+=…)

Soal-soal terkait :

<https://pandaoj.com/problem/SUM>

<https://pandaoj.com/problem/BNPCHS2009J>

<https://pandaoj.com/problem/IDEAFUSE16G>

<http://www.spoj.com/problems/DIVSUM/>

1. Sorting

Sorting merupakan proses pengurutan , intinya begitu wkwk , simpenya kalo di cpp sudah ada stl sort(), tinggal gunakan saja~ .

Soal-soal terkait :

<https://pandaoj.com/problem/BNPCHS2010QB>

<https://pandaoj.com/problem/BNPCHS2014D>

1. Deret

Jangan lupakan rumus deret ! (terutama deret aritmatika) , karena cukup banyak soal yg memerlukan penghitungan dengan deret terutama 2 rumus deret aritmatika ini :

Un=a+(n-1)\*b

Sn=(n\*(2a+(n-1)\*b)/2

Soal terkait :

<https://pandaoj.com/problem/MAGIC>

<https://pandaoj.com/problem/JC8D> (+binary search)

1. Ad-hoc , string related

Ad-hoc string merupakan salah satu tipe soal yg sebenarnya wajib untuk dikuasai, hanya saja untuk soal string sebenarnya mudah-mudah sulit , mudah nya biasanya perintahnya obvious dan sangat jelas suruh ngapain , tetapi cukup kerepotan ketika implement nya , terkhusus untuk soal string yang mana kita harus pandai2 dengan yang namanya parsing string (pecah-pecah string) , untuk terbiasa dalam menyelesaikan soal string, latihan intens sangat diperlukan

Soal terkait :

<https://pandaoj.com/problem/CONVERSION>

<https://pandaoj.com/problem/ASDASD>

<https://pandaoj.com/problem/INC2013A>

<https://pandaoj.com/problem/BNPCHS2009B>

<https://pandaoj.com/problem/INC2015B>

1. Searching
   1. Binary Search

Binary search merupakan algoritma untuk pencarian yang mana dia akan membagi 2 range terus menerus hingga mendapat jawaban yang diinginkan. Algoritma ini jauh lebih cepat daripada linear search , dimana dalam linear seach kita harus mengiterasi seluruh isi array untuk mendapatkan jawaban yang diinginkan, algoritma ini bekerja dengan syarat data sudah terurut.

Contoh template binary search:

<http://ideone.com/cpm9sq>

kode diatas merupakan salah satu penerapan binary search untuk keperluan mencari batas bawah atau lower\_bound() , pada penerapannya binary search tidak hanya mencari data di dalam data , ada juga yang namanya binary search the answer , yaitu jawaban yang kita tebak kemudian kita simulasikan permasalahan , apakah permasalahan dapat diselesaikan dengan jawaban yang kita tebak? Singkatnya seperti itu.

Soal-soal terkait :

<https://pandaoj.com/problem/WATER2>

<https://pandaoj.com/problem/DOUBLE>

<https://pandaoj.com/problem/BNPCHS2014C> (binary search the answer or find O(1) formula )

<https://pandaoj.com/problem/TREMENDOUS>

1. DFS / BFS

Untuk DFS/BFS nanti menyusul ya , karena penjelasannya agak ribet ,

* BFS(Breath First Search) memakai queue
* DFS(Depth First Search) memakai stack

Void dfs{

1. Dijkstra , adjacent silice :

Vector<int> array[100];

Jadi dalam array ada vector, 2 Dimensi, jadi mudah untuk push dan manipulasi apa saja yang terkoneksi dengan arraynya.