Dự tuyển olympic tin học

Thư viện STL

Giới thiệu về thư viện STL

- Standard Template Library
  - Trở thành chuẩn của C++ từ phiên bản C++98
  - Sử dụng kĩ thuật template
  - "`using namespace std;`"
- Gồm 4 thành phần chính:
  - Các thuật toán cơ bản (algorithms)
  - Các vật chứa cơ bản (containers)
  - Các con chạy (iterators – con trỏ được đóng gói bởi template)
  - Các hàm cơ bản (functions)
- Dùng được STL cực kỳ lợi thế khi lập trình thi đấu vì hầu hết các thuật toán và xử lý cơ bản đã được cung cấp
  - Nhưng hiểu được STL thì không hề đơn giản

Ví dụ: hàm find_if

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

bool isFive(int x) {
    return x == 5;
}

int main() {
    vector<int> a = {1, 2, 3, 4, 5};
    cout << find_if(begin(a), end(a), isFive) - begin(a);
}
```

Hàm tìm kiếm của thư viện STL C++

- Thư viện <algorithm>
- Tìm tuyến tính:
  - find: tìm vị trí xuất hiện của giá trị trong dãy
  - find_if: tìm vị trí xuất hiện của giá trị trong dãy theo điều kiện
  - search: tìm vị trí xuất hiện của đoạn con trong dãy
- Tìm nhị phân (dãy tăng dần):
  - binary_search: kiểm tra xem có phần tử trong dãy hay không
  - lower_bound: trả về vị trí của phần tử đầu tiên không bé hơn phần tử cần tìm
  - upper_bound: trả về vị trí của phần tử đầu tiên lớn hơn phần tử cần tìm
  - includes: kiểm tra xem đoạn con có xuất hiện trong dãy hay không

Cài đặt sắp xếp ở thư viện STL C++

- Thư viện <algorithm>
- sort: sắp xếp (tăng dần) một đoạn, sử dụng introsort
- stable_sort: sắp xếp ổn định (tăng dần) một đoạn, sử dụng mergesort
- partial_sort: sắp xếp phần đầu của đoạn theo thứ tự tăng dần, sử dụng khi ta chỉ cần lấy vài phần tử nhỏ nhất
- nth_element: sắp xếp đoạn và trả về phần tử thứ n
- partition: chia đoạn làm đôi theo tiêu chí của hàm chọn
- make_heap: tạo một heap từ đoạn đã cho

Nhiều hàm hữu ích khác

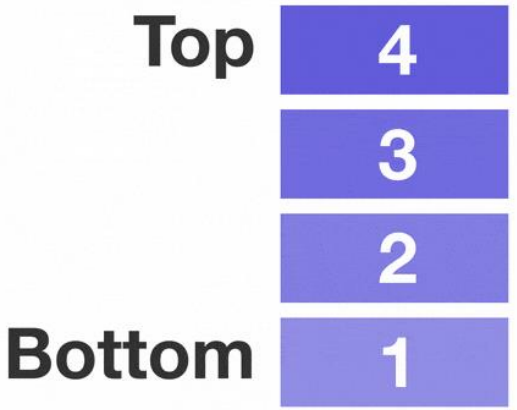
- min_element / max_element / minmax_element
- any_of / all_of / none_of
- count / count_if
- equal / mismatch
- reverse / rotate / shuffle / next_permutation
- copy / transform / generate / fill
- replace / replace_if / remove / remove_if
- for_each
- reduce / accumulate
- …

Các cấu trúc dữ liệu thông dụng

- Mảng tĩnh: array<T, size>
- Mảng động: vector<T>
- Hàng đợi hai đầu: deque<T>
- Danh sách liên kết đôi: list<T>
- Danh sách liên kết đơn: forward_list<T>

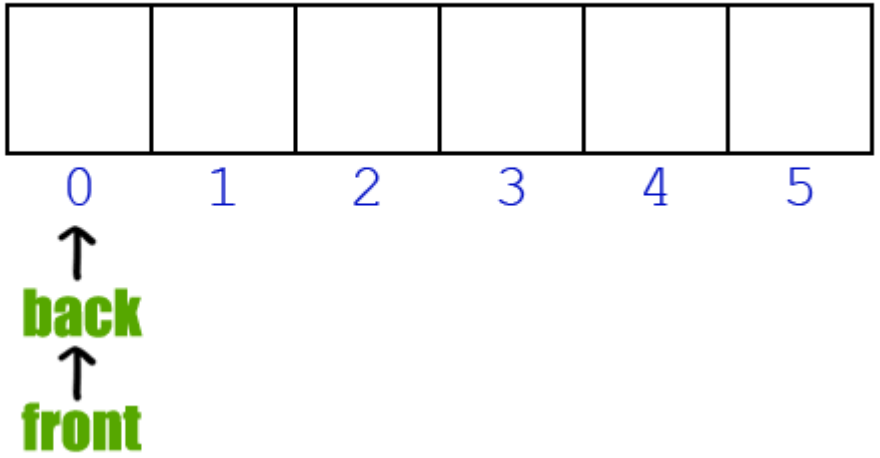
Stack

- Ngăn xếp
- LIFO: last-in, first-out
- Thường được cài đặt dựa trên list, vector, array
- Thao tác cơ bản:
  - Thêm vào (push): đặt vào cuối
  - Lấy ra (pop): lấy ra phần tử ở cuối



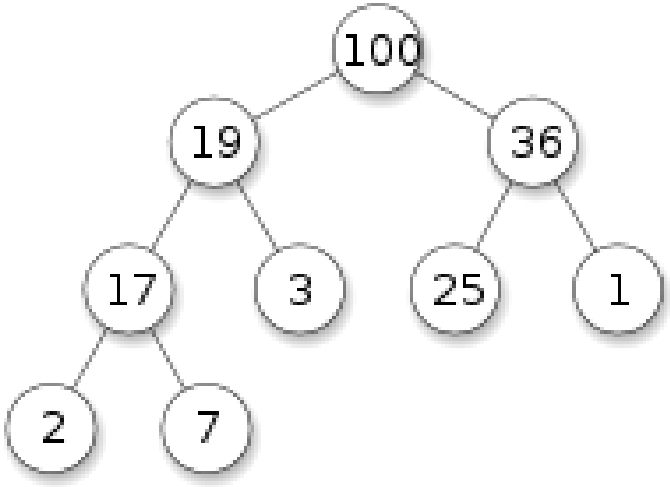
Queue

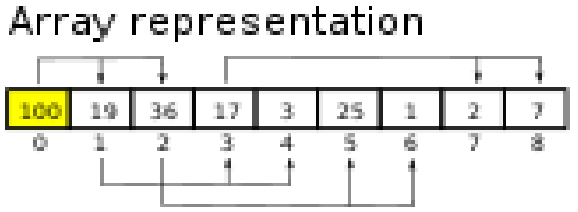
- Hàng đợi
- FIFO: first-in, first-out
- Thao tác cơ bản: enqueue / dequeue
  - Thêm vào (push): thêm vào cuối
  - Lấy ra (pop): lấy phần tử ở đầu
- Dạng hai đầu: Deque



Heap

- Đống
- Còn gọi là priority queue (hàng đợi ưu tiên)
  - Dữ liệu tổ chức dạng heap, thứ tự giảm dần
  - Thêm vào (push): tự đặt phần tử vào vị trí phù hợp trong heap
  - Lấy ra (pop): lấy phần tử lớn nhất
- Cấu trúc sử dụng trong heap sort





Set

- Tập hợp
- Các phần tử phải khác nhau
- Thường cài đặt trên red-black tree hoặc hash table
- Vài kiểu dữ liệu cùng loại:
  - multiset
  - unordered_set
  - unordered_multiset

Map

- Ánh xạ
- Từ điển
- Cho phép ánh xạ từ một khóa (key) tới giá trị (value)
- Vài kiểu dữ liệu cùng loại:
  - multimap
  - unordered_map
  - unordered_multimap