Bài 5. Cấu trúc Vector

Nội dung

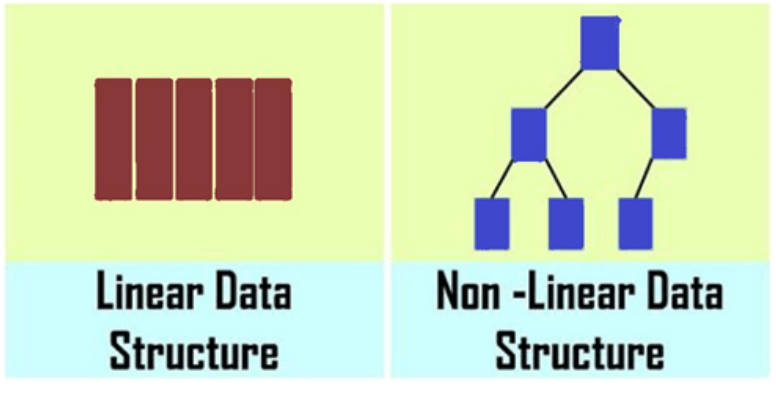
I. Cấu trúc dữ liệu tuyến tính, phi tuyến
II. Kiểu dữ liệu vector
III. Cấu trúc vector trong C++
IV. Các ứng dụng của Vector
V. Cài đặt vector bằng mảng
VI. Phát triển mảng
VII. Bộ lặp - Iterator

I. Cấu trúc tuyến tính, phi tuyến

- **C**ấu trúc dữ liệu **tuyến tính** là một cấu trúc mà các phần tử dữ liệu được liên kết với nhau tuần tự. Một phần tử chỉ kết nối với một phần tử trước và một tử sau.
- Bằng cách tổ chức như vậy, ta có thể duyệt qua các phần tử của nó trong một lần chạy.
- **Một số ví dụ:**
  - Danh sách (lists)
  - Vector, chuỗi (vectors sequences)
  - Danh sách kiểu ngăn xếp, danh sách kiểu hàng đợi (stack, queue)

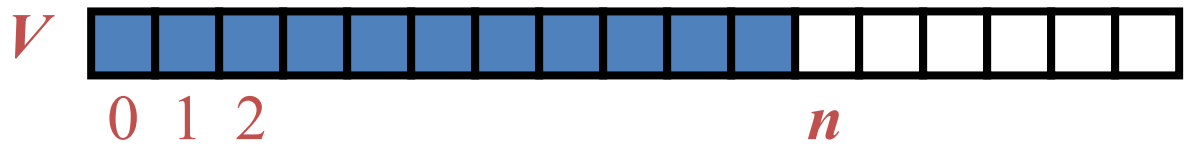
- **C**ấu trúc dữ liệu **phi tuyến** là một cấu trúc mà các phần tử dữ liệu được liên kết với nhau thứ tự tuyến tính. Một phần tử có thể kết nối với nhiều phần tử trước, sau.

- **Một số ví dụ:**
  - Cây (tree)
  - Đồ thị (graph)



II. Kiểu dữ liệu Vector (Vector ADT)

- **Kiểu dữ liệu trừu tượng Vector là sự mở rộng của khái niệm mảng. Vector là một mảng lưu trữ một dãy các đối tượng với số lượng tùy ý.**



- Một phần tử có thể được truy cập, chèn thêm hoặc loại bỏ đi khi biết chỉ số của nó.

- Khi thực hiện các thao tác trên có thể xảy ra lỗi nếu chỉ số của phần tử không chính xác (Vd, chỉ số âm)

III. Cấu trúc Vector trong C++

- **Khai báo**

  vector<T> vec_name(n);

  vector<T> vec_name(n, value);

  T arr = {a_1,a_2, ..,a_n };

  vector<T> vec_name(arr, arr + n);

  Hoặc

  vector<T> vec_name(arr, arr + sizeof(arr)/sizeof(T));

III. Cấu trúc Vector trong C++

Các nhóm thao tác của Vector

1. Modifiers
2. Iterators
3. Capacity
4. Element access

III. Cấu trúc Vector trong C++

Modifiers

- assign(int size, int value);
- pop_back()
- insert(position, value);
- erase(position);
- erase(start-position, end-position);
- emplace(position, element);
- emplace_back()
- emplace_back(value);
- swap(vector);
- clear()

Iterators

1. vector<int>::iterator vec_it;
   - begin()
   - end()

2. vector<int>::reverse_iterator vec_it;
   - rbegin()
   - rend()

III. Cấu trúc Vector trong C++

Capacity

- size()
- max_size()
- capacity()
- resize(n)
- resize(int n, int value);
- empty()
- shrink_to_fit()
- reserve(n):

Element access

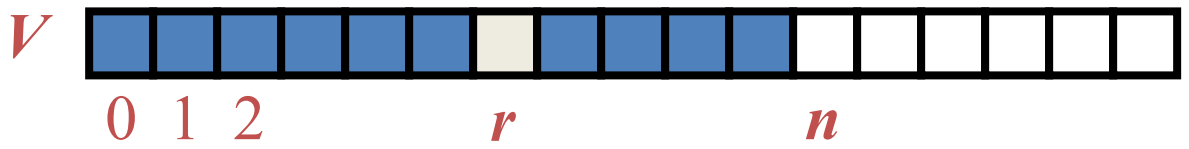
- at(position);
- data()
- front()
- back()
- []

IV. Các ứng dụng của Vector

- **Ứng dụng trực tiếp**
  - Lưu trữ tập hợp các đối tượng (cơ sở dữ liệu đơn giản)
- **Ứng dụng gián tiếp**
  - Cấu trúc dữ liệu bổ trợ cho các thuật toán
  - Thành phần của các cấu trúc dữ liệu khác

V. Cài đặt Vector bằng mảng

- Sử dụng mảng *V* có kích thước *cap (capacity)*
- Một biến *num* (number) lưu trữ kích thước của vector (số phần tử được lưu trữ)
- Phép toán reference operator[] được thực hiện trong thời gian $O(1)$ bằng việc trả lại *V*[*r*]



Tóm lại

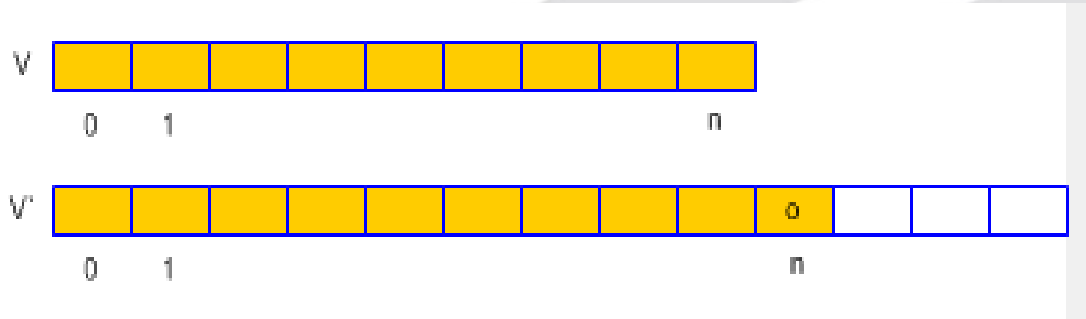
- **Cài đặt Vector bằng mảng:**
  - Không gian sử cho cấu trúc dữ liệu là $O(n)$
  - Các phép toán *size*, *empty*, *at* và *replace* chạy trong thời gian $O(1)$
  - *insert* và *erase* chạy trong thời gian $O(n)$
- **Với phép toán *insert*, khi mảng đầy sẽ dẫn đến ngoại lệ, để tránh trường hợp này chúng ta thay mạng hiện tại bằng mảng lớn hơn**
- **Để thêm phần tử vào khi mảng đầy thì ta cần làm như thế nào?**

VI. Phát triển mảng

- Khi thực hiện phép toán. Nếu mảng đầy sẽ dẫn đến xảy ra lỗi. Để có thể thêm phần tử đó vào ta phải mở rộng mảng.
- **Làm thế nào để mở rộng mảng?**
  - Chiến lược phát triển theo hằng số:Tăng thêm kích thước mảng theo một hằng số ***c***
  - Chiến lược gấp đôi:Tăng gấp đôi số phần tử hiện có của mảng

Thêm phần tử vào cuối

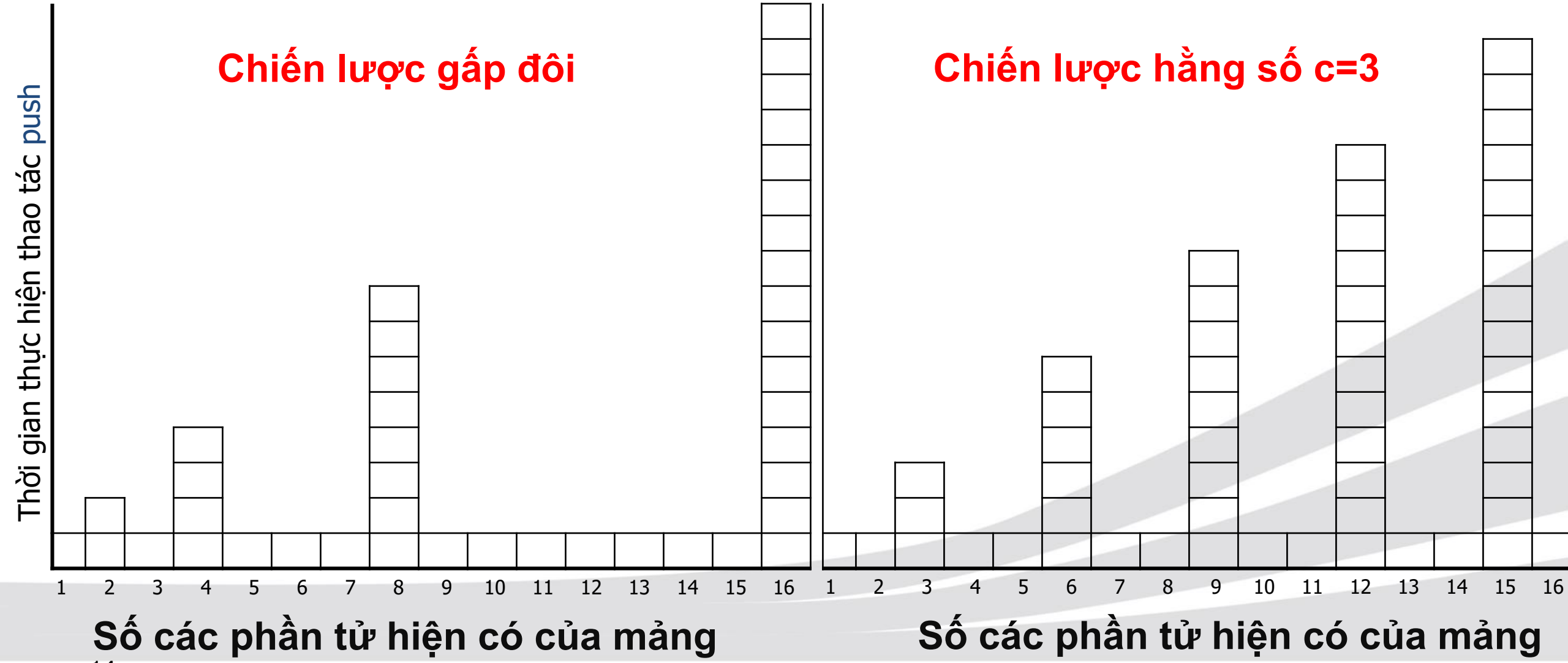
```
Algorithm push_back( o)
   if n = V.N – 1 then
      A ← Tạo mảng mới có kích thước
…
      for i ← 0 to n-1 do
          A[i] ← V[i]
      delete V
      V ← A
      V[n] ←O
      n ← n+ 1
```



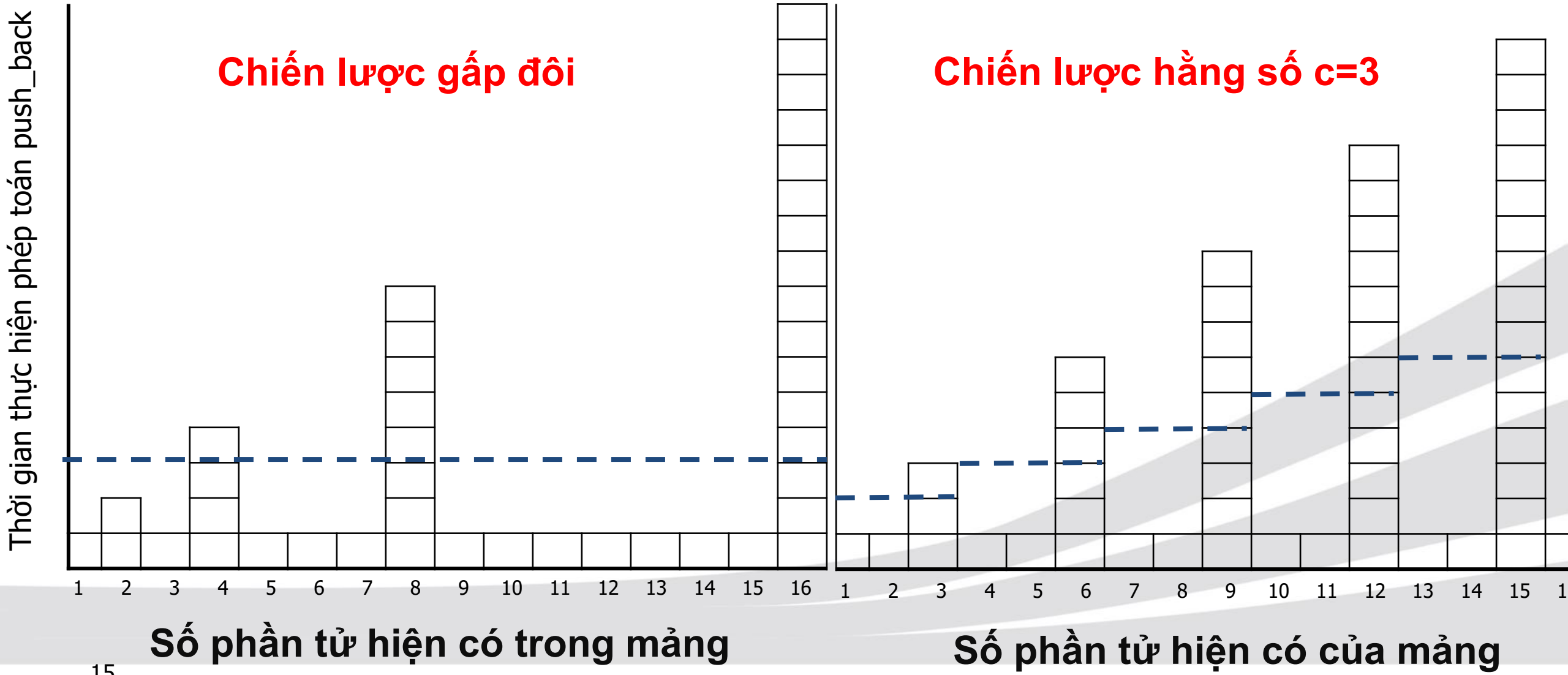
So sánh hai chiến lược

- Ta so sánh chiến lược phát triển theo hằng số và chiến lược gấp đôi bằng cách phân tích tổng thời gian $T(n)$ cần thiết để thực hiện thao tác push_back một dãy n phần tử vào mảng.
- Chúng ta thực hiện bắt đầu với mảng có 1 phần tử
- Và đi xác định thời gian trung bình khi push_back một phần tử vào mảng là $T(n)/n$

Thời gian thực hiện đưa một dãy các phần tử vào mảng bằng cách sử dụng



Thời gian thực hiện đưa một dãy các phần tử vào mảng bằng cách sử dụng chiến lược



Phân tích chiến lược phát triển theo hằng số

- Chúng ta thay thế mảng $k = n/c$ lần
- Tổng thời gian $T(n)$ của phép toán push_back n phần tử vào mảng tương ứng là

$$n + c + 2c + 3c + 4c + \ldots + kc =$$

$$n + c(1 + 2 + 3 + \ldots + k) =$$

$$n + ck(k + 1)/2$$

- Trong đó c là một hằng số, $T(n)$ là $O(n + k^2)$, hay là $O(n^2)$
- Vậy thời gian trung bình phải trả cho phép toán push là $O(n)$

Phân tích chiến lược gấp đôi

- Chúng ta thay thế mảng ***k*** = log2 ***n*** **lần**

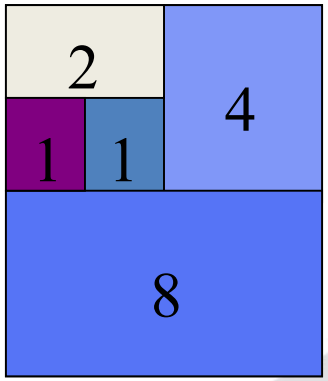
  - **Tổng thời gian thực hiện phép toán push n phần tử vào mảng là $T(n)$ và tương ứng là:**

$$n + 1 + 2 + 4 + 8 + \ldots + 2^k =$$

$$n + 2^{k+1} - 1 = 3n - 1$$

  - **$T(n)$ là $O(n)$**
  - **Thời gian trung bình phải tra cho phép toán push_back một phần tử mảng là $O(1)$.**

Mô tả bằng hình học

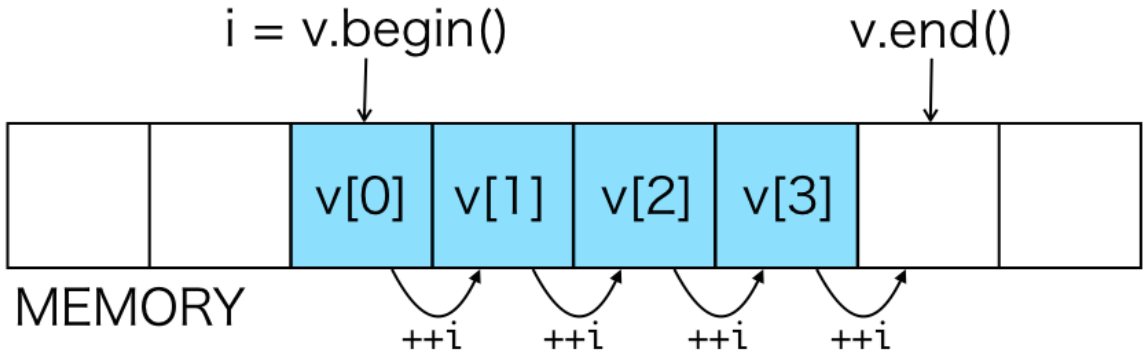


Một số lưu ý khi cài đặt bằng mảng

- Khi sử dụng mảng để cài đặt Vector thì việc sử dụng ngôn ngữ có khả năng dễ dàng cấp phát bộ nhớ là rất quan trọng.
- Trong một số ngôn ngữ, mảng không thể mở rộng được sau khi nó đã được tạo ra.
- Các ngôn ngữ thủ tục được chia thành 2 lớp ngôn ngữ dựa vào khia cạnh này:
  - Các ngôn ngữ như là: Ada, Algol, C và JAVA cho phép xác định kích thước mảng vào thời gian chạy chương trình khi mảng được tạo ra. Như vậy, mảng có thể mở rộng tùy ý.
  - Các ngôn ngữ như là: Pascal, Modula-2 and Fortran thì rất hạn chế, nó yêu cầu kích thước của mảng phải được xác định khi dịch chương trình.
  - Nếu không thể thay đổi động thì khi cài đặt các cấu trúc dữ liệu bằng mảng phải có phương pháp mở rộng mảng.

VII. Iterator – Bộ lặp

- Trong các ADT không hỗ trợ phép tìm duyệt các phần tử của nó.
- Bộ lặp được sử dụng để duyệt lần lượt các phần tử của các ADT như: Vector, List, Tree,…
- Tại một thời điểm đối tượng bộ lặp cung cấp cho người dùng một phần tử của đối tượng ADT theo thứ tự nào đó đã được xác định.
- Tại một thời điểm có thể có nhiều đối tượng bộ lặp trên cùng một đối tượng ADT




```
v[k] == *(v.begin()+k)
```

Ví dụ sử dụng bộ lặp của lớp vector trong c++

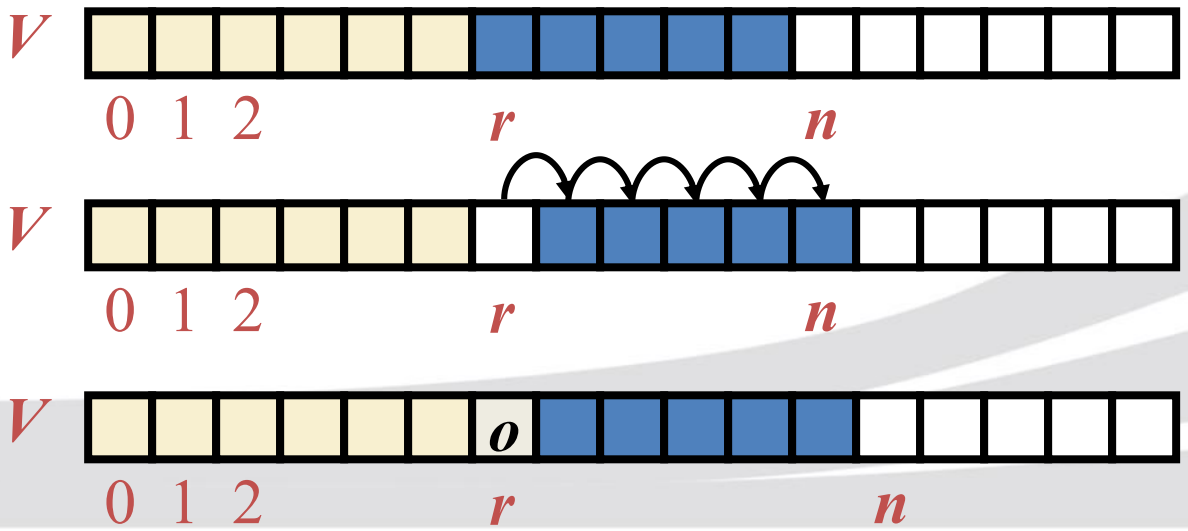
```
#include"bits/stdc++.h"
using namespace std;
int main()
{
    vector<int> V(7,6);
    cout<<"\nDuyet:";
    vector<int>::iterator it;
    for(it=V.begin();it!=V.end();++it)
        cout<<*it<<"\t";
}
```

Cấu trúc bộ lặp

- **Thuộc tính**
  - Một thuộc tính lưu địa chỉ của thuộc tính quản lý các phần tử của ADT sử dụng bộ lặp
  - Một thuộc tính lưu địa chỉ của phần tử hiện tại của ADT sử dụng bộ lặp
- **Phương thức**
  - thao tác gán =
  - thao tác tham lấy giá trị tại vị trí nó trỏ tới *it
  - thao tác ++it và it++

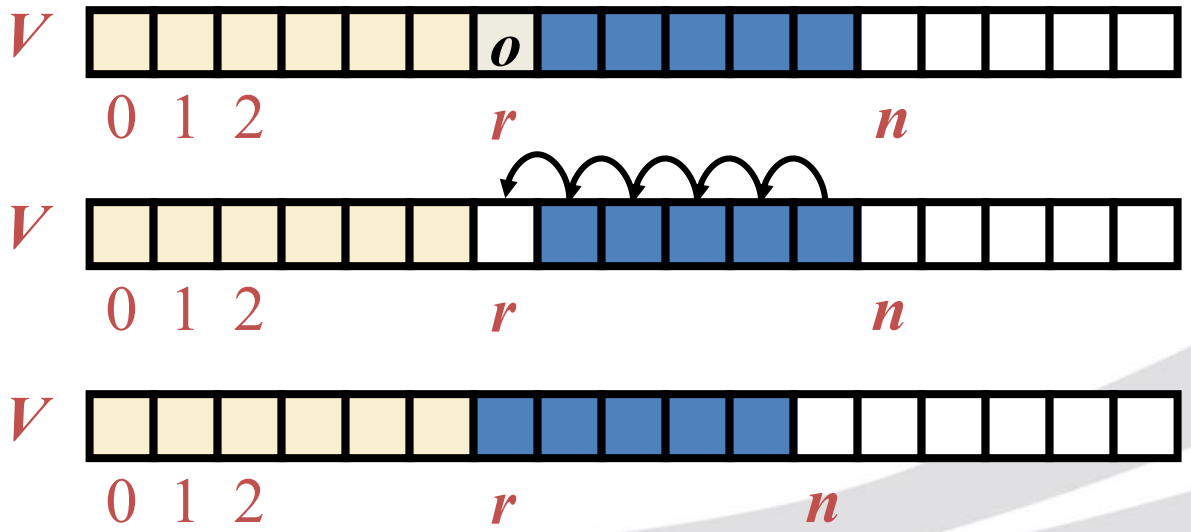
Chèn thêm phần tử

- iterator insert (const_iterator position, const value_type& val)
- Chúng ta cần tạo một ô mới có chỉ số r bằng cách đẩy n-r phần tử từ *V*[*r*], …, *V*[*n* - 1] về sau 1 vị trí
- Trong trường hợp xấu nhất (*r* = 0), phép toán thực hiện trong thời gian *O*(*n*)



Loại bỏ phần tử

- Phép toán iterator erase (const_iterator position) chúng ta cần đẩy n - *r* - 1 phần tử từ *V*[*r* + 1], …, *V*[*n* - 1] về trước một vị trí
- Trong trường hợp xấu nhất (*r* = 0), phép toán thực hiện trong thời gian *O*(*n*)



Cài đặt Vector bằng C++

Tham khảo code tại : https://ideone.com/LyeviE

Tham khảo:
https://topdev.vn/blog/vector-trong-c/#vector-tu-ghi-nho-do-dai-cua-minh

Ứng dụng

I. Xây dựng lớp Vector và lớp bộ lặp của nó

II. Sử dụng lớp Vector và lớp bộ lăp của lớp vector xây dựng chương trình có các chức năng sau:

1. Chèn 1 phần tử vào vector
2. Xóa 1 phần tử của vector
3. Thay thế một phần tử của vector
4. Lấy giá trị của một phần tử của vector
5. In danh sách các phần tử hiện có trong vector

Các phần tử lưu vào vector là các số thực

Ví dụ tính n! với n <= 1000

- Ta dùng vector lưu các đảo ngược chữ số của n!
- Ví dụ 7!=5040 thì ta lưu 0 4 0 5 để tính khi xuất ra thì đảo ngược lại

Code minh họa n!

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    long long n,memory=0;
    cin>>n;
    vector<int> V(1,1);
    for(int i=2;i<=n;i++)
    {
        memory=0;
        for(auto it=V.begin();it!=V.end();it++)
        {
            memory+=*it*i;
            *it=memory%10;
            memory/=10;
        }
        while(memory) {V.push_back(memory%10);memory/=10;}
    }
    for(auto it=V.rbegin();it!=V.rend();it++) cout<<*it;
}
```

Hết