# Vector in C++

- Vector 是 C++ 提供的 sequence data container, 本質上是能夠改變大小的陣列
- 如同陣列一樣， vector 使用連續的空間來儲存資料，並且能以索引值在 O(1) 時間內存取
- Vector 內部，使用動態配置的空間儲存資料。
  - 有需要時，會重新配置新的(更大的)空間，並把資料搬過去。
    (實際使用的空間，在執行時期可能會改變。)

# Vector in C++

- 重新配置記憶體、並搬移全部的資料，是很耗時的動作。
  - Vector 內部使用 doubling technique 改善效率問題
  - 平均的存取時間仍然為 O(1)
    O(1) Amortized access time
- 平均來說，使用多不到一倍的空間與時間，換取可動態調整的方便性

# Vector in C++

- 宣告方式

```
vector< data type >  var;
```

- 使用方式
  - 當成陣列使用 –
    以 [] 索引值存取, 需注意不可超過範圍
  - 使用成員函式存取

# Vector in C++

- 常用成員函式
  - size() 傳回 vector 內的資料數量
  - push_back(data)

    將 data 加至陣列的結尾, 若空間已滿, 則自動調整、動態配置新空間
  - pop_back()

    移除陣列最後的元素

# Deque in C++

- Deque 是 C++ 提供的一種 sequence data container, 內部為 doubly-linked list
- Deque 使用動態配置的節點空間來儲存空間，可以在 O(1) 的時間在 list 的頭尾兩邊新增/刪除 資料
- 無法有效率 access list 中間的元素，需要使用 iterator 遍歷整個 list
  - Iterator 為指向 C++ STL 容器裡的資料的指標, 大部份情況下,可當成指標來使用)

# Deque in C++

- 常用成員函式
  - size() 傳回 deque 內的資料數量
  - push_back(data), push_front(data)
  - pop_back(), pop_front()
  - back(), front()

    傳回 list 後端/前端的資料的 reference

# Vector 與 Deque 之比較

| | Vector | Deque |
|---|---|---|
| Insert / Delete at One End | O(1) | O(1) |
| Insert / Delete at Both Ends | O(N) | O(1) |
| Insert / Delete in the Middle | O(N) | O(1), when pointer is provided |
| Random Access | O(1) | O(N) |

# 使用 vector 或 deque 建圖

- 考慮以下的問題
  - 給定一個任意的圖 (graph)
  - 問題： 是否能從點 u 走到點 v ？

# 使用 vector 或 deque 建圖

- 考慮以下的問題
  - 給定一個任意的圖 (graph)
  - 問題： 是否能從點 u 走到點 v ？
  - 使用 vector 或 deque 儲存每個點的「鄰居列表」 Adjacency List
  - 從 u 開始 DFS 探索 (遞迴走迷宮)

# Iterate over Containers

- 使用 iterator 遍歷 container 裡的資料
  - container 的成員函式

    begin() - 傳回指向第一個元素的 iterator

    end() - 傳回**指向container結尾**的 iterator
    **(注意:非最後一個元素)**

  - 使用作用於 iterator 的 ++, -- 運算子來做移動