Date . .

(Queue)

First In First Out (FIFO)

佇列 三 排隊

ADT queue operations

Create an empty queue 建構

Destroy a queue 解構

. Determine whether a queue is empty 是否為空

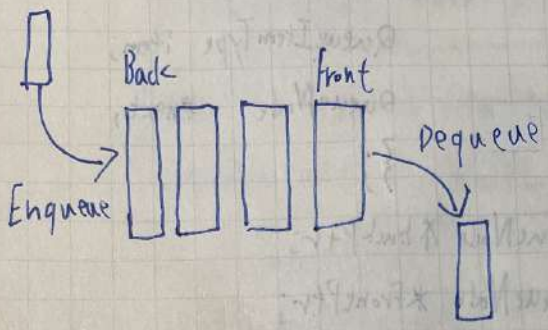
Add a new item to the queue 新增

Remove the item that was added earliest 移除。

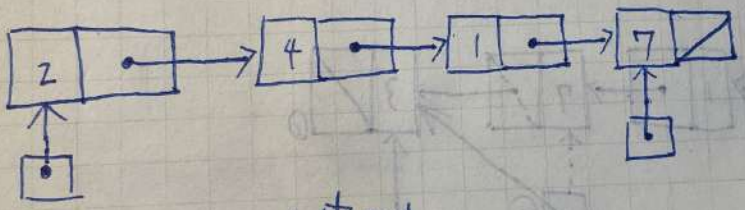
Retrieve the item that was added earliest 擷取

⁂ dequeue() 擷取後移除。

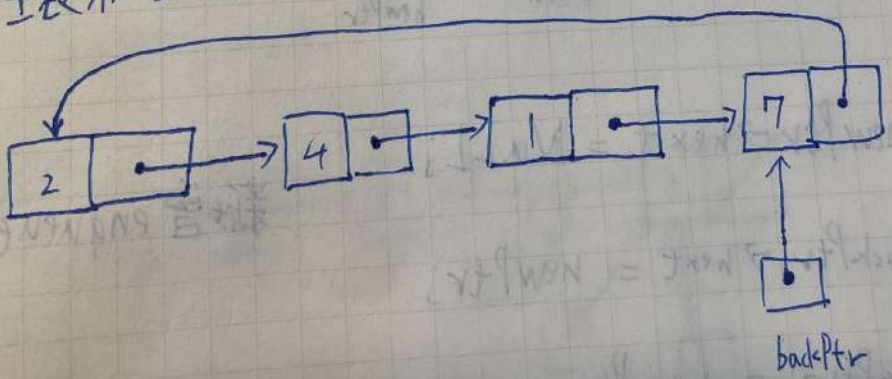
Implementations of the ADT Queue



一般鏈結串列

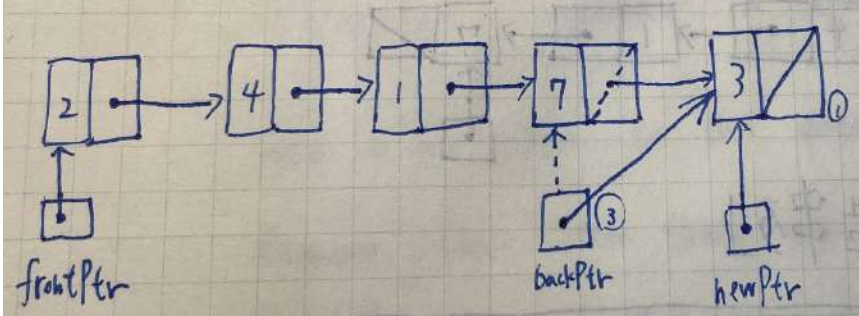


環狀鏈結串列



```
private:  struct QueueNode {
              QueueItemType item;
              QueueNode    *next;
              };
          QueueNode *backPtr;
          QueueNode *frontPtr;
```

自行實作基本形態



1, newPtr->next = NULL;

2, backPtr->next = newPtr;

3, backPtr = newPtr

新增 enqueue

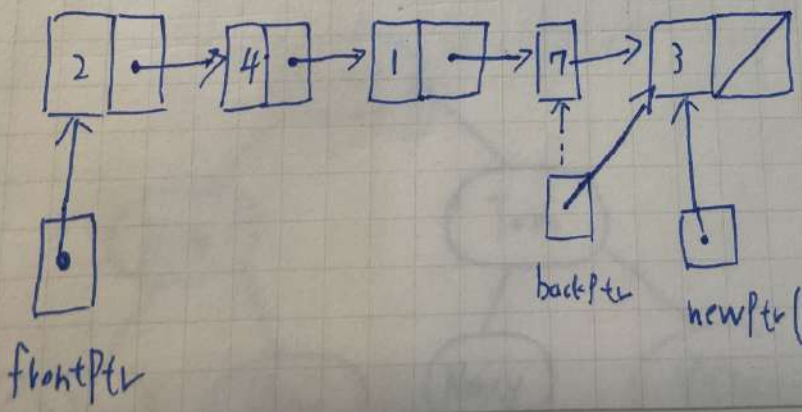
Date

```
Void enqueue(const QueueItemType& newItem)
{ QueueNode *newPtr = new QueueNode;
  newPtr->item = newItem;      新節點內容
  newPtr->next = NULL;         反指標
  if (isEmpty())
    frontPtr = newPtr
  else backPtr->next = newPtr ②
  backPtr = newPtr; ③
}
```

1, newPtr->next = NULL;

2, backPtr->next = newPtr;

3, backPtr = newPtr;



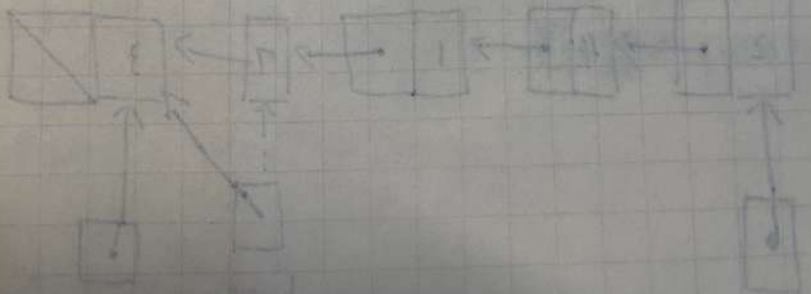
Date

希爾排序法

```
Void shellSort (int A[], int n)
{ for (int h = n/2; h > 0; h = h/2)
  for (int unsorted = h; unsorted < n; ++unsorted)
  { int loc = unsorted;
    int nextItem = A[unsorted];
    for (; (loc >= h) && (A[loc-h] > nextItem); loc=loc-h)
      A[loc] = A[loc-h];

  A[loc] = nextItem;
  }

}
```

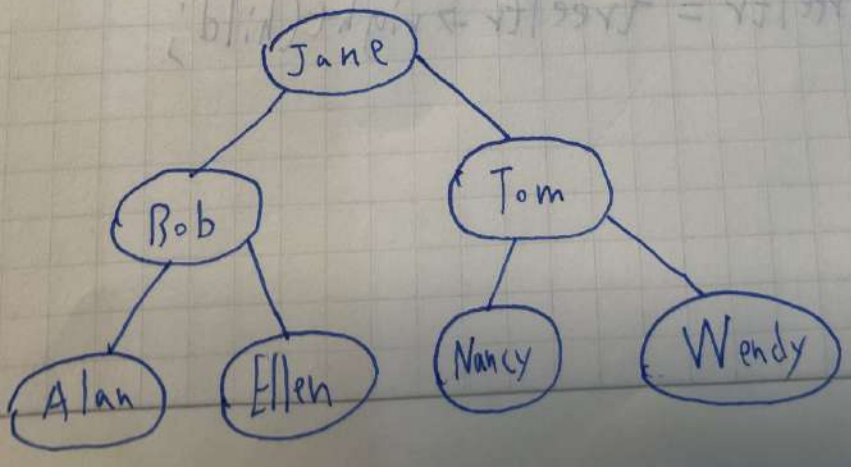


Date . .

BST – Deletion

三個可能

- X is a leaf 葉節點
  - Set the pointer in X's parent to NULL
- X has only one child 一個小孩的節點
  - Let X's parent adopt X's only child
- X has two children 兩個小孩的節點
- X has two children?



```
inorderTraversal (binaryTree root)
{ binaryTree   treePtr = root;
  nodeStack    aStack;

while (! aStack.empty() || (treePtr != NULL))
{ while (treePtr != NULL)
  {  aStack.push (treePtr);
     treePtr = treePtr -> leftChild;
  }
aStack.pop (treePtr);
cout << treePtr -> data << endl;
treePtr = treePtr -> rightChild;
}
}
```