

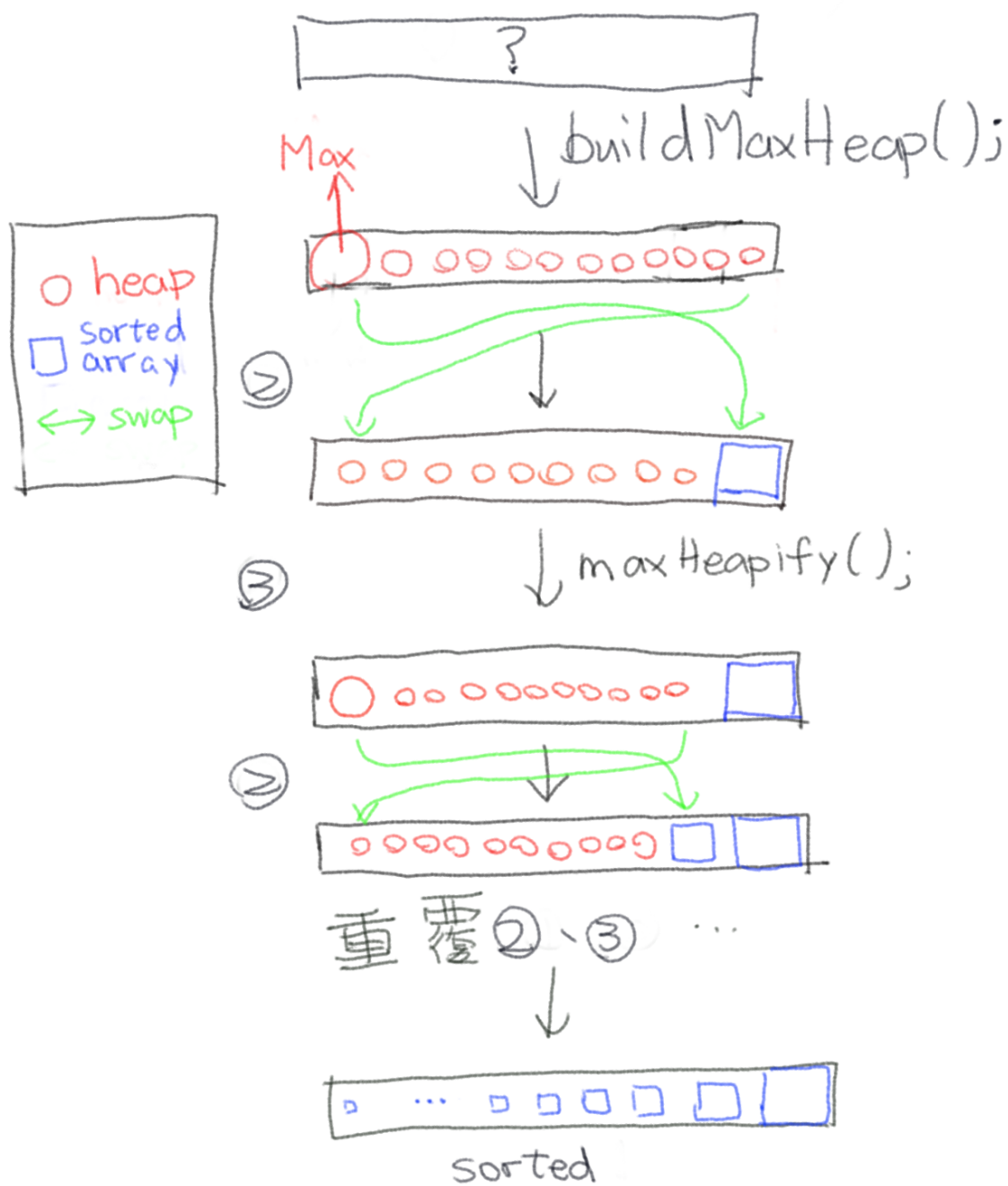
## 資料結構

```
/* max data size to store*/
#define MAX_DATA_SIZE 200
/*max ID size, 9 characters + null character*/
#define MAX_ID_SIZE 10
/*max name size, 40 characters + null character*/
#define MAX_NAME_SIZE 41
/*employee資料結構*/
typedef struct employee{
    /*ID*/
    char id[MAX_ID_SIZE];
    /*姓名*/
    char name[MAX_NAME_SIZE];
    /*每月工作時數*/
    unsigned monthly_working_hour;
    /*時薪*/
    unsigned hourly_salary;
}Employee;
```

- 藉由父母節點(parent node)與其左右子女節點(children node)序號的規律性可用陣列來表示二元樹。
  - $PARENT(Node(i)) = Node(i / 2)$
  - $LEFT\_CHILD(Node(i)) = Node(2i)$
  - $RIGHT\_CHILD(Node(i)) = Node(2i + 1)$

## 演算法模型

- 先實作「於兩子女樹皆為 max heap 的前提下」遞迴地將可能違背 heap property 的父母節點(parent node)與子女節點(children node)互換向下移位(shift down?)以促成 max heap 的 maxHeapify()函式。
- 實作自最後一個非葉節點(non-leaf node)的節點到根節點(root node)依序呼叫 maxHeapify()函式使得整個資料結構皆成為 max heap 的函式 buildMaxHeap()。
- 實作 heapSort(), 先呼叫 buildMaxHeap()使資料來源陣列成為 max heap, 然後將此時為最大數值的根節點(root node)跟陣列中索引(index)值最大的資料互換並遞減當前 heap 的大小最後再透過 maxHeapify()修復 heap property, 重複動作就能將資料由大至小一一排序。



## 已完成項目

1. 自認為使用者體驗比較好的開啟檔案提示(List\_directory\_files/listDirectoryFiles.h)
2. 透過標準 C++ 函式庫提供的物件讀取檔案資料 ( 理論支援正確處理 Windows("\r\n") 跟 Unix("\n") 作業系統平台的行結尾(E.O.L.)序列 )  
(portableEOLalgorithm/portableEOLalgorithm.h)。
3. heapsort 演算法(Sorting\_algorithm/heapsort.h):
  - heapSortEmployee() ← 包含 delete() ?
  - maxHeapifyEmployee() ← 包含 shiftDown() ?
  - buildMaxHeapEmployee()
  - 執行時間似乎不是  $O(n)$ ... 具體應該要用什麼演算法還沒有查出來。
4. 用 assertion 作資料正確性的檢查(io.cpp)。

## 測試結果

1. 於 Unix 平台下 istream::getline() 函式讀取 Windows/MS-DOS 平台行結尾序列會殘留 '\r' 字元於讀取到的字串末端 ( 印出該字串時會輸出多餘的空行 )。
2. Inline 函式似乎不能被其他檔案 include ( 只有 file 的 scope ? )。

## 已知問題

1. 由於資料結構中工作時數跟時薪使用 unsigned 整數型態, io.cpp 中讀取檔案使用 istream::getline() → atoi(String.c\_str()) 無法藉由判斷是否為非負來判斷資料是否存在異常數值。雖然直接將資料結構更改為 signed 型態可以解決這個問題不過目前是朝向改用 C++ ifstream 的函式檢查 fail bit 的方向處理。

## 平台及相容性

- 原始程式碼: 相容於 GNU G++、Visual C++ 編譯器。
- 可執行檔: 相容於 Linux & Windows & Macintosh ( 理論 ) 於 x86 CPU 架構。