qsort() / bsearch() in stdlib

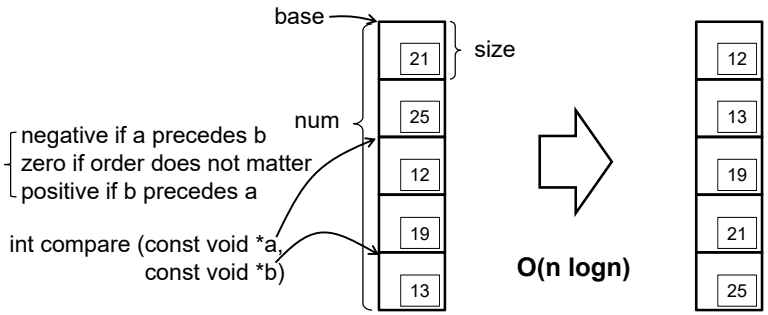
Pei-yih Ting

Sorting an array of data

stdlib

- void **qsort**(void* **base**, size_t **num**, size_t **size**,
  int (***compare**)(const void*,const void*));

quick sort: divide the data into approximately a half at each step, and sort them individually



A simple int array

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
int compare(const void *ptr_a, const void *ptr_b);
int main() {
   int i, n=6, values[] = {40, 10, 100, 90, 20, 25};
   qsort(values, n, sizeof(int), compare);
   for (i=0; i<n; i++) printf("%d ", values[i]);
   return 0;
}
int compare(const void *ptr_a, const void *ptr_b) {
    int *ptr_a1 = (int *)ptr_a, *ptr_b1 = (int *)ptr_b;
    if (*ptr_a1 < *ptr_b1)
        return -1;
    else if (*ptr_a1 == *ptr_b1)
        return 0;
    else
        return 1;
}
```

```
int compare(const void *ptr_a, const void *ptr_b) {
    return *(int *)ptr_a - *(int *)ptr_b;
}
```


A simple double array

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
int compare(const void *ptr_a, const void *ptr_b);
int main() {
   int i, n=6; double values[] = {40.3, 10.2, 100.9, 90.1, 20.2, 25.4};
   qsort(values, n, sizeof(double), compare);
   for (i=0; i<n; i++) printf("%6.1f ", values[i]);
   return 0;
}
int compare(const void *ptr_a, const void *ptr_b) {
    double *ptr_a1 = (double *)ptr_a, *ptr_b1 = (double *)ptr_b;
    if (*ptr_a1 < *ptr_b1)
        return -1;
    else if (*ptr_a1 == *ptr_b1)
        return 0;
    else
        return 1;
}
```

相等的比對不太容易成功，但是也不會造成什麼問題，運算得到的兩個浮點數一般來說不是大於就是小於

```
int compare(const void *ptr_a, const void *ptr_b) {
    return *(double *)ptr_a - *(double *)ptr_b;
}
```

浮點數轉換為**整數** ⇒ 小數部份**完全捨去**

An array of int[2]

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
int compare(const void *a, const void *b);
int main() {
   int i, n=6, values[][2] = {{40,17}, {10,19}, {100,21}, {40,15}, {20,11}, {25,16}};
   qsort(values, n, sizeof(int[2]), compare);
   for (i=0; i<n; i++) printf("%d,%d ", values[i][0], values[i][1]);
   return 0;
}
```

40,17 10,19 100,21 40,15 20,11 25,16 → 20,11 25,16 40,15 40,17 10,19 100,21

Swap-based sort: Unstable

```
int compare(const void *a, const void *b) {
   int *a1 = (int *)a, *b1 = (int *)b;
   if (a1[0] < b1[0])
      return -1;
   else if (a1[0] == b1[0])
      return 0;
   else
      return 1;
}
```

using the **first** field as the key for comparison

or using the **second** field as the key for comparison

```
int compare(const void *a, const void *b) {
   return ((int *)a)[1] – ((int *)b)[1];
}
```


int[2] array with multi-field key

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
int compare(const void *a, const void *b);
int main() {
   int i, n=6, values[][2] = {{40,17}, {10,19}, {40,21}, {40,15}, {20,11}, {25,16}};
   qsort(values, n, sizeof(int[2]), compare);
   for (i=0; i<n; i++) printf("%d,%d\n", values[i][0], values[i][1]);
   return 0;
}
```

40,17 10,19 40,21 40,15 20,11 25,16 → 10,19 20,11 25,16 40,15 40,17 40,21

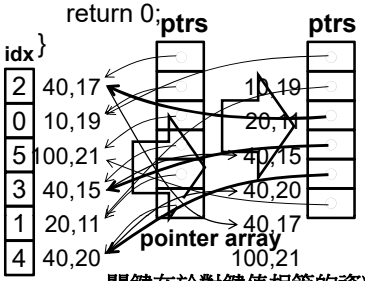
```
int compare(const void *a, const void *b) {
   int *a1 = (int*)a, *b1 = (int*)b;
   return a1[0]==b1[0] ? a1[1]-b1[1] : a1[0]-b1[0];
}
```

```
int compare(const void *a, const void *b) {
   long long a1 = *(int*)a, b1 = *(int*)b,
             a2 = *((int*)a+1), b2 = *((int*)b+1);
   return (a1<<32+a2) - (b1<<32+b2);
}
```


Stable Sorting

```
#include <stdio.h>      /* printf
#include <stdlib.h>     /* qsort
int compare(const void *a, co
int main() {
   int i, n=6, values[][2] = {{40
   int *ptrs[6]; for (i=0; i<n; i++
   qsort(values, n, sizeof(int[
   qsort(ptrs, n, sizeof(int*), c
   for (i=0; i<n; i++) printf("%d
   for (i=0; i<n; i++) printf("%d
   return 0;
}
```

```
void swapArray(int *a, int *b, int len) {
   int t, i;
   for (i=0; i<len; i++) t=a[i], a[i]=b[i], b[i]=t;
}
void rearrange(int values[][2], int *ptrs[]) {
   int idx[6];
   for (i=0; i<n; i++)
      idx[(ptrs[i]-values[0])/2] = i;
   for (i=0; i<n-1; i++) {
      swapArray(values[i], ptrs[i], 2);
      idx[(ptrs[i]-values[0])/2] = idx[i];
      ptrs[idx[i]] = ptrs[i];
   }
   for (i=0; i<n; i++)
      printf("%d,%d\n", values[i][0], values[i][1]);
}
```



Stable

when **(int **)a == **(int **)b, the addresses *(int *)a and *(int *)b are never equal

might or might not swap them

想要把原本陣列依照順序 (in place 排好的話，需要第二個陣列 idx 紀錄每一個 values[i] 在 ptrs 陣列裡的位置，當 ptrs[i] 與 values[i] 陣列元素互換的時候，修改 idx[ptrs[i]-values[0])/2] = idx[i]; 相等時，判斷結果不相等

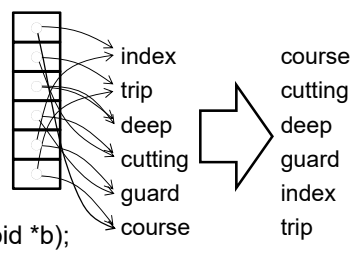
Unstable

when **(int **)a == **(int **)b, the addresses *(int *)a and *(int *)b are never equal

the in-place quick sort algorithm

關鍵在於對鍵值相等的資料定義出相對順序，原本位置比較前面的是比較小的

Sorting array of strings char *[n]



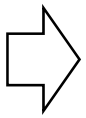

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
#include <string.h>     /* strcmp */
int compare(const void *a, const void *b);

int main() {
   int i, n=6;
   char *strs[] = {"index", "trip", "deep", "cutting", "guard", "course"};
   qsort(strs, n, sizeof(char*), compare);
   for (i=0; i<n; i++) printf("%s\n", strs[i]);
   return 0;
}
```

```
int compare(const void *a, const void *b) {
   char **ptr_a = (char **)a, **ptr_b = (char **)b;
   return strcmp(*ptr_a, *ptr_b);
}
```

An array of user-defined struct

```
#include <stdio.h>     /* printf */
#include <stdlib.h>    /* qsort */
#include <string.h>    /* strcmp */
int compare(const void *a, const void *b);

struct Data {
    char name[20];
    int age;
};

int main() {
    int i, n=6; struct Data values[] = {{"John",17}, {"Bob",19}, {"Kathy",21},
                                        {"Melody",15}, {"Jim",11}, {"Anthony",16}};
    qsort(values, n, sizeof(struct Data), compare);
    for (i=0; i<n; i++) printf("%s,%d\n", values[i].name, values[i].age);
    return 0;
}
```

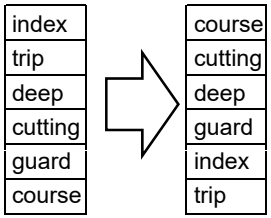
John,17
Bob,19
Kathy,21
Melody,15
Jim,11
Anthony,16



Anthony, 16
Bob, 19
Jim,11
John,17
Kathy,21
Melody,15

```
int compare(const void *a, const void *b) {
    struct Data *ptr_a = (struct Data *)a,
                *ptr_b = (struct Data *)b;
    return strcmp(ptr_a->name, ptr_b->name);
}
```


Sorting array of strings char [n][8]




```
#include <stdio.h>     /* printf */
#include <stdlib.h>    /* qsort */
#include <string.h>    /* strcmp */
typedef int (*CFP)(const void *, const void *);
int compare(const void *a, const void *b);

int main() {
    int i, n=6;
    char strs[][8] = {"index", "trip", "deep", "cutting", "guard", "course"};
    qsort(strs, n, sizeof(char[8]), (CFP)strcmp);
    for (i=0; i<n; i++) printf("%s\n", strs[i]);
    return 0;
}
```

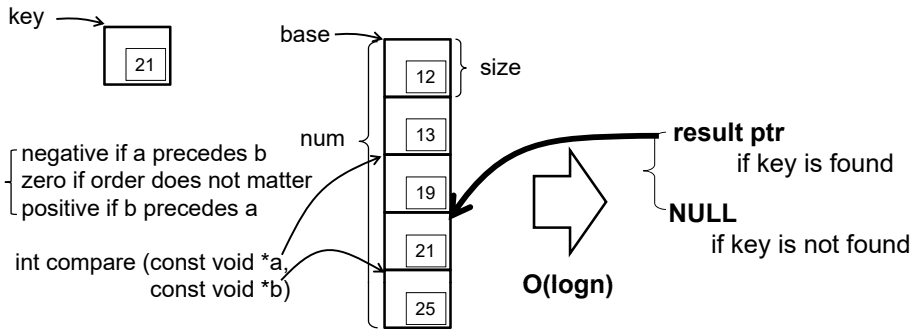
```
int compare(const void *a, const void *b) {
    char *ptr_a = (char *)a, *ptr_b = (char *)b;
    return strcmp(ptr_a, ptr_b);
}
```


Searching data in a sorted array

stdlib

- const void* **bsearch**(const void ***key**, void* **base**, size_t **num**, size_t **size**, int (***compare**)(const void*,const void*));

Binary search : divide the sorted data into approximately a half at each step, and eliminate one of them with a single comparison



if multiple entries in the array have this key, **any one of them** could be returned

Self-crafted Binary Search

- Problem: **If target is not found**, bsearch returns NULL.
  - What if we want to know the immediate element that follows the target?

```
int binarySearch(int target, int data[], int left, int right) {
    int mid;
    while (left <= right) {
        mid = (left+right)/2;
        if (target > data[mid]) left = mid + 1;
        else if (target < data[mid]) right = mid - 1;
        else return mid;
    }
    return left; // if target∉data[] : target < data[L]
}
return -1;  // if target∈data[] : data[L]==target data[L-1] < target
}
```

HDU 1075
What Are You Talking About

- http://acm.split.hdu.edu.cn/showproblem.php?pid=1075
- **Problem**: Ignatius is so lucky that he met a Martian yesterday. But he didn't know the language the Martians use. The Martian gives him a history book of Mars and a dictionary when it leaves. Now Ignatius want to translate the history book into English. Can you help him?

Sample Input:

```
START
from fiwo
hello difh
mars riwosf
earth fnnvk
like fiiwj
END
START
difh, i'm fiwo riwosf.
i fiiwj fnnvk!
END
```

Sample Output:

```
hello, i'm from mars.
i like earth!
```


Still Unstable sorting with qsort()

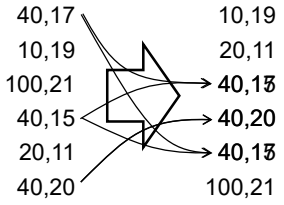
```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */
int compare(const void *a, const void *b);

int main() {
   int i, n=6, values[][2] = {{40,17}, {10,19}, {100,21}, {40,15}, {20,11}, {40,20}};
   qsort(values, n, sizeof(int[2]), compare);
   for (i=0; i<n; i++) printf("%d,%d\n", values[i][0], values[i][1]);
   return 0;
}
```

```
int compare(const void *a, const void *b) {
   int order = (int*)a - (int*)b;
   return order==0 ? (int*)a - (int*)b : order;
}
```

when *(int *)a == *(int *)b
the difference of these two addresses is never zero

in-place quick sort 執行過程中
key 相等的資料還可能和其它資料比對並且交換位置

key 相等時, 原本相對前面的一直維持在前面



Unstable
Stable