主題: Sorting

- 基礎
- 應用
- 作業與自我挑戰

基礎

- What is sorting ?
- qsort
- Make use of index tables

What is sorting ?

- 將給定的資料按照特定的順序排列好
  - 由小到大
  - 由大到小
- 例：
  - 1, 7, 9, 5, 3
  - 由小到大：1, 3, 5, 7, 9
  - 由大到小：9, 7, 5, 3, 1

Stable sort

- 大小相同的 items，sort 完後會依照 input 時的順序排列
  - input: 73 95 62 81 73 84
  - stable sort: 62 73 73 81 84 95

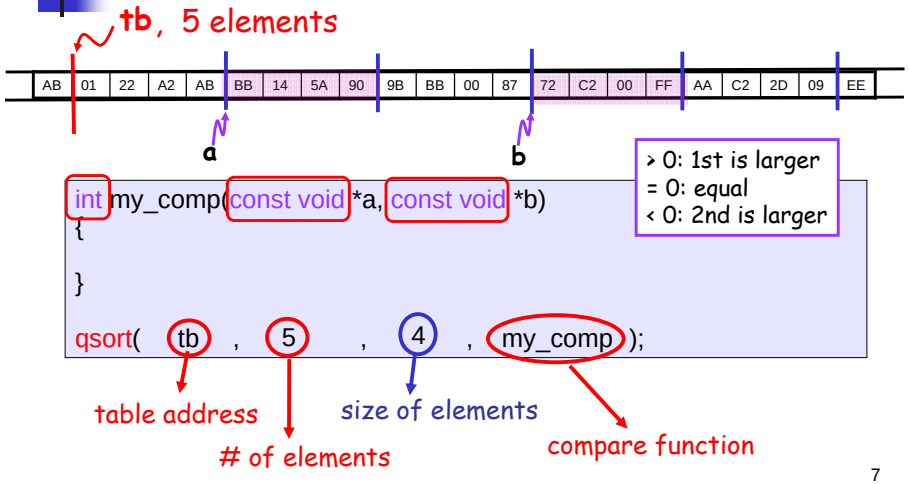
常見的 sorting 演算法

- bubble sort
- merge sort
- quick sort
- integer sort

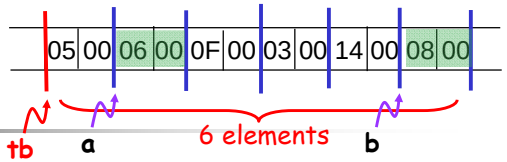
qsort

- C 內建的 sort function (<stdlib.h>)
  - quick sort
  - worst case: $O(n^2)$
  - average case: O(n lg n)
  - 一般來說都已經夠用
- 只需自己寫一個 compare function

Usage: sort an array tb in memory



Example




```
short int tb[] = {5, 6, 15, 3, 20, 8};

int comp_func(const void *a, const void *b)
{
   short int c, d;
   c = * (short int *) (a);
   d = * (short int *) (b);   } get contents (two integers)
   if (c > d) return (1);
   else if (c == d) return (0);
   else return (-1);
}

qsort(tb, sizeof(tb) / sizeof(tb[0]), sizeof(tb[0]), comp_func);
```

> 0: 1st is larger
= 0: equal
< 0: 2nd is larger

table address # of elements size of elements compare function

Example

```
char tb[][10] = {"Test", "OK", "Hello", "Book", "BBS", "C"};

int comp_func(const void *a, const void *b)
{
   char *c, *d;
   c = (char *) (a);
   d = (char *) (b);
   return (strcmp(c, d));
}

qsort(tb,  sizeof(tb) / sizeof(tb[0]),   sizeof(tb[0]), comp_func);
```

two pointers to strings

strcmp ???

Example

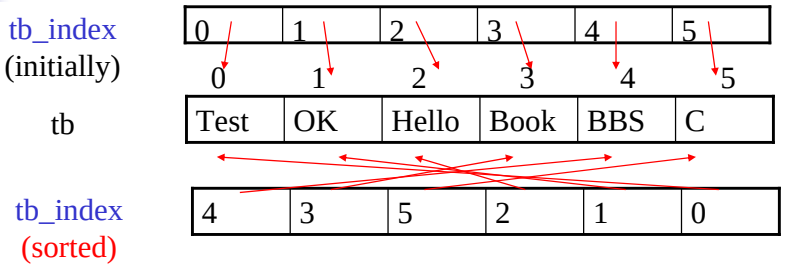
```
#include <stdlib.h>
int num[5];

int comp_func(const void *a, const void *b)
{  int c, d;
   c = * (int *)(a);
   d = * (int *)(b);
   return (c – d);
}

int main(void)
{
   read_input(num);
   qsort(num, sizeof(num)/sizeof(num[0]), sizeof(num[0]), comp_func);
   /*  qsort(num, 5, sizeof(num[0]), comp_func);  */
}
```

may overflow!!!

Make use of index tables



- Usages:
  - Avoid swapping large items (e.g., strings, structures)
  - Sort by different keys (one index table for a key)
  - Make sorting stable

Example: make qsort stable

```
char tb[][10] = {"Hello", "OK", "Test", "Hello", "Book", "C"}; tb_size = 6;
int tb_index[] = {0, 1, 2, 3, 4, 5};

int comp_func (const void *a, const void *b)
{  int i1, i2, x;
   i1 = * (int *) (a);
   i2 = * (int *) (b);
   x = strcmp(tb[i1], tb[i2]);
   if (x != 0) return(x);
   return(c - d);
}

qsort(tb_index, tb_size, sizeof(int), comp_func);
```

two indices

由 index (id) 決定

overflow ???

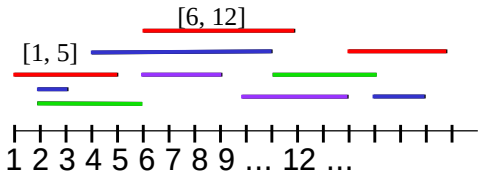
index table

應用

- 應用一: Intersections of intervals
- 應用二: A.10125 Sumsets

應用一: Intersections of intervals

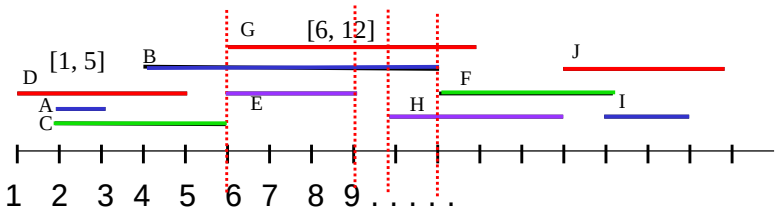
- 給 n 個 intervals I_1, I_2, ..., I_n
  - Each interval [a, b] represents the set {a, a+1, a+2, ..., b}
- 將這些 intervals 分堆，每一堆中的 intervals 要 disjoint
- 請問最少要分成幾堆?
- $n \le 10^6$; $1 \le a \le b \le 10^8$



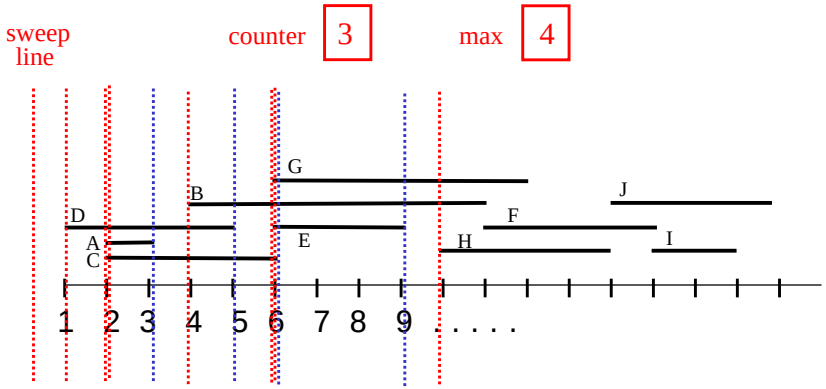
- 類題: AF.2326 Moving Tables
  (**Remark**: This problem is much easier, since n ≤ 200.)

Observations

- minimum number of subsets
  = maximum number of overlapping intervals
- same intervals between two consecutive endpoints



Solution: plane-sweep



Solution (cont.)

- Step 1. Sort all endpoints x_i
  - primary-key: x-coordinates
  - secondary-key: left/right endpoints

  } multiple keys
- Step 2. Scan the endpoints x_i from left to right
  - Initially, counter = 0
  - left endpoint: counter + 1
  - right endpoint: counter – 1
- Output: the maximum value of counter
- Time: O(n lg n) ~ 10^7

Data items with multiple fields

Array of structures (records)

- struct ENDPOINT{int x; char lr;};
- ENDPOINT endpoint[10];

endpoint[3].x

endpoint[3].lr

	0	1	2	3	4	5	6	7	8	9
endpoint	1	5	2	3	2	6	4	11	6	9
	0	1	0	1	0	1	0	1	0	1

Parallel arrays

- int x[10]; char lr[10];

	0	1	2	3	4	5	6	7	8	9
x	1	5	2	3	2	6	4	11	6	9
lr	0	1	0	1	0	1	0	1	0	1

Sorting with multiple keys

primary-key: x; secondary-key: lr

Implementation 1. qsort on an array of structures

```
int comp_func (const void *a, const void *b)
{  ENDPOINT e1, e2;
   e1 = * (ENDPOINT *) (a); e2 = * (ENDPOINT *) (b); } two endpoints
   if (e1.x > e2.x)  return (1);
   else if (e1.x < e2.x) return (-1);                  } primary-key
   else return ((int) e1.lr - (int) e2.lr);            } secondary-key
        /* primary keys are the same
}
qsort(endpoint, 10, sizeof(ENDPOINT), comp_func);
```

array of structures

return (e1.lr - e2.lr) ???

Implementation 2. qsort on an array of structures by using an index table

```
int endpoint_index[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int comp_func (const void *a, const void *b)
{  int i1, i2; ENDPOINT e1, e2;
   i1 = * (int *) (a);  i2 = * (int *) (b);   } two indices
   e1 = endpoint[i1]; e2 = endpoint[i2];   } two endpoints
   if (e1.x > e2.x)  return (1);
   else if (e1.x < e2.x) return (-1);                  } primary-key
   else return ((int) e1.lr - (int) e2.lr);            } secondary-key
        /* primary keys are the same
}
qsort(endpoint_index, 10, sizeof(int), comp_func);
```

index table

Implementation 3. qsort on parallel arrays by using an index table

```
int endpoint_index[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int comp_func (const void *a, const void *b)
{  int i1, i2;
   i1 = * (int *)(a); i2 = * (int *)(b);   } two indices
   if (x[i1] > x[i2]) return (1);
   else if (x[i1] < x[i2]) return (-1);     } primary-key
   else return ((int) lr[i1] - (int) lr[i2])  } secondary-key
      /* primary keys are the same
}
qsort(endpoint_index, 10, sizeof(int), comp_func);
```

index table

Remark. Stable sort is a special case of sorting with multiple keys

應用二: A.10125 Sumsets

- 給 n 個整數，請找出最大的一個數字 d 使得 a + b + c = d，其中 a b c d 是這堆數字中四個不同的數字
  - 每個數字的範圍在 -536870912 ~ 536870911
  - $n \le 10^3$
  - 所有數字都不相同

Solution

- 測試所有 (a, b, c, d) 的組合: work or not ???
- 將等式換成 a + b = d – c
  - a + b 和 d – c 各只有1000*1000 種可能
  - 把所有可能的 x = a + b 都算出來存 X
  - 把所有可能的 y = d - c 都算出來存 Y
  - 檢查有沒有 x = y ?
- Note: Record (a, b) for each x and record (c, d) for each y
  - 一組 valid (a, b, c, d) 必會產生一組 x = y
  - 一組 x = y 未必能產生一組 valid (a, b, c, d) Why???

Example

- Input
  - 1, 2, 3, 5, 7, 12
- 任兩個數字和的所有可能 X
  - 3, 4, 5, 6=1+5, 7, 8, 8, 9, 10=3+7, 12, 13, 14, 15, 17, 19
- 任兩個數字差的所有可能 Y
  - 11=12-1, 10=12-2, 9=12-3, 7=12-5, 5=12-7
    6=7-1, 5=7-2, 4=7-3, 2=7-5,
    4=5-1, 3=5-2, 2=5-3, 2=3-1, 1=3-2, 1= 2-1

How to find x = y?

- Check all (x, y) pairs ???
- 利用 binary search
  - sort X
  - for every y, find all x = y by using binary search
    - check whether (a, b, c, d) are distinct
  - the largest d ???
  - time complexity ???
- hashing

Remark

- Assume that a number can appear several times in the input
  - 大小相同的整數視為不同的數字
- For each y = c – d
  - at most 10^6 (a, b) with x = y
  - at most 10^3 contributed by c
  - at most 10^3 contributed by d
- Time: $10^6 \times \lg 10^6 \times 10^3 \sim 10^{10}$
- Can you solve this problem in 10^8 operations ???

作業與自我挑戰

- 作業
  - 練習題
    - A.10125 Sumsets
      http://uva.onlinejudge.org/external/101/10125.html
  - 挑戰題
    - A.812 Trade on Verweggistan
      http://uva.onlinejudge.org/external/8/812.html
- 自我挑戰
  - AF2002-2481 Silly Sort (Min cost swap sort)
  - AF2001-2239 Professor Monotonic's Networks (0/1 principle)
  - A. 10587 Mayor's Posters
    http://uva.onlinejudge.org/external/105/10587.html

- 其他題目
  - A.612 DNA Sorting (compute inversion)
    http://uva.onlinejudge.org/external/6/612.html
  - A.10008 What's Cryptanalysis
    http://uva.onlinejudge.org/external/100/10008.html
  - A.10057 A Mid-summer Night's Dream
    http://uva.onlinejudge.org/external/100/10057.html
  - H.89.1 對對碰