HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

C BASIC

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

C Basic

SORTING AND APPLICATIONS (PART 1)

ONE LOVE. ONE FUTURE.

CONTENT

- Sorting a sequence of integers (P.05.11.01)
- Sorting a sequence of strings (P.05.11.02)
- Ranking learning ability (P.05.11.03)

SORTING A SEQUENCE OF INTEGERS (P.05.11.01)

- Given a sequence of integers $a_1, a_2, \ldots, a_n$. Sorting the sequence in non-decreasing order
- Data
  - Line 1: An integer n ($1 <= n <= 100000$)
  - Line 2: Values $a_1, a_2, \ldots, a_n$ where($1 <= a_i <= 1000000$)
- Result
  - The sorted list

stdin	stdout
5 4 3 6 5 1	1 3 4 5 6

SORTING A SEQUENCE OF INTEGERS - PSEUDOCODE

- Applying heap sort

```
Heapify(a, i, n){
  L = 2*i; R = 2*i+1; maxIdx = i;
  if L <= n and a[L] > a[maxIdx] then maxIdx = L;
  if R <= n and a[R] > a[maxIdx] then maxIdx = R;
  if maxIdx != i then {
     swap(a[i], a[maxIdx]); Heapify(maxIdx, n);
  }
}
BuildHeap(){
  for i = n/2 downto 1 do Heapify(i, n);
}
HeapSort(){
  BuildHeap();
  for i = n downto 2 do {
     swap(a[1], a[i]); Heapify(1, i-1);
  }
}
```

SORTING A SEQUENCE OF STRINGS (P.05.11.02)

- Given a sequence of strings S_1, S_2, . . ., S_n. Sort the sequence in lexicographic order
- Data
  - Line 1: An positive integer n (1 <= n <= 100000)
  - Line 2: Values of S_1, S_2, . . ., S_n where the lengths of the string are between 1 and 100
- Result
  - Each line a string in the sorted sequence

stdin	stdout
10 O0001 Z002 R003 R00004 P05 P00006 T0007 X08 N09 I010	I010 N09 O0001 P00006 P05 R00004 R003 T0007 X08 Z002

SORTING A SEQUENCE OF STRINGS - PSEUDOCODE

- Using an array of pointers, each pointer points to an array of characters (dynamically allocated)
- When swapping 2 elements, only swap 2 pointers (do not use the string copy function).
- Heap sort algorithm is applied

```
Heapify(a, i, n){
  L = 2*i; R = 2*i+1; maxIdx = i;
  if L <= n and a[L] > a[maxIdx] then maxIdx = L;
  if R <= n and a[R] > a[maxIdx] then maxIdx = R;
  if maxIdx != i then {
     swap(a[i], a[maxIdx]); heapify(maxIdx, n);
  }
}
BuildHeap(){
  for i = n/2 downto 1 do Heapify(i, n);
}
HeapSort(){
  BuildHeap();
  for i = n downto 2 do {
     swap(a[1], a[i]); Heapify(1, i-1);
  }
}
```

RANKING LEARNING ABILITY (P.05.11.03)

- There is a list of students that need to be ranked by score, each student has 2 information fields:
  - <studentID>: student id: a string with a length between 1 and 10
  - <grade>: score (positive integer)
- Know that students' scores are different. Calculate each student's position in the ranking (the number of students in the list with a smaller score)
- Data
  - Line 1: A positive integer *n* (1 <= *n* <= 100000)
  - Line i+1 (i = 1, 2, ..., n): <studentID> and <grade> of i^{th} student
- Result
  - Each line <studentID> and the rank in the ranking (The lines are sorted in lexicographic order by student number)

stdin	stdout
5 S000003 3 S000002 6 S000005 5 S000004 10 S000001 8	S000001 3 S000002 2 S000003 0 S000004 4 S000005 1

RANKING LEARNING ABILITY - PSEUDOCODE

- Sort the list of students in ascending order of scores using the sorting algorithm to accumulate the list $S_1, S_2, \ldots, S_n$.
- Then the position of student S_i is i (i = 1, 2, …, n)
- Sort the student list in lexicographic order by student number and print the results.

```
Struct Student {
   ID; // Student ID
   grade; // Score
   pos; // Position
}
```

```
Heapify(S, i, n){
   L = 2*i; R = 2*i+1; maxIdx = i;
   if L <= n and S[L].grade > S[maxIdx].grade then maxIdx = L;
   if R <= n and S[R].grade > S[maxIdx].grade then maxIdx = R;
   if maxIdx != i then {
      swap(S[i], S[maxIdx]); Heapify(maxIdx, n);
   }
}
BuildHeap(){
   for i = n/2 downto 1 do Heapify(i, n);
}
HeapSort(){
   BuildHeap();
   for i = n downto 2 do {
      swap(a[1], a[i]); Heapify(1, i-1);
   }
}
```

RANKING LEARNING ABILITY - PSEUDOCODE

- Sort the list of students in ascending order of scores using the sorting algorithm to accumulate the list $S_1, S_2, \ldots, S_n$.
- Then the position of student S_i is i (i = 1, 2, …, n)
- Sort the student list in lexicographic order by student number and print the results.

```
Struct Student {
   ID; // Student ID
   grade; // Score
   pos; // Position
}
```

```
Main() {
   Read the student list S[1], S[2], . . ., S[n];
   HeapSort();
   for i = 1 to n do S[i].pos = i-1;
   Sort S[1], . . ., S[n] by Student ID;
   for i = 1 to n do {
      print(S[i].ID, ' ', S[i].pos);
   }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !