Taller ProgComp: Track Básico

Algoritmos de Ordenamiento

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

April 16, 2024

PROBLEMA

Definición

Dado un arreglo de n enteros, queremos ordenarlo según un método de comparación. Por ejemplo, de menor a mayor.

4	2	3	5	8	2

Problema

Definición

Dado un arreglo de n enteros, queremos ordenarlo según un método de comparación. Por ejemplo, de menor a mayor.

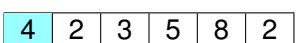
4	2	3	5	8	2

2	2	3	4	5	8

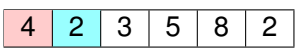
Part I

Fuerza Bruta

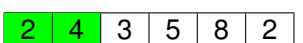
Insertion Sort



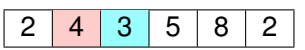
Insertion Sort



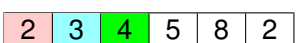
Insertion Sort



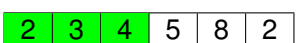
Insertion Sort



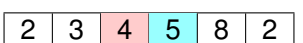
Insertion Sort



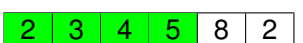
Insertion Sort



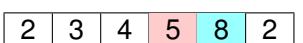
Insertion Sort



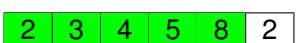
Insertion Sort



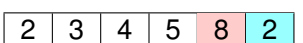
Insertion Sort



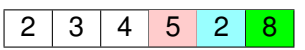
Insertion Sort



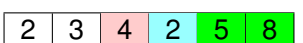
Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort

Insertion Sort

Insertion Sort

Insertion Sort - Código

```cpp
void insertion_sort(vector< int > &nums) {
  for(int i = 1; i < nums.size(); i++) {
    int x = nums[i];
    int j = i;
    while(j > 0 && nums[j - 1] > x) {
      nums[j] = nums[j - 1];
      j--;
    }
    nums[j] = x;
  }
}
```

Insertion Sort - Código

```cpp
void insertion_sort(vector< int > &nums) {
  for(int i = 1; i < nums.size(); i++) {
    int x = nums[i];
    int j = i;
    while(j > 0 && nums[j - 1] > x) {
      nums[j] = nums[j - 1];
      j--;
    }
    nums[j] = x;
  }
}
```

¿Cuál sería la complejidad?

Insertion Sort - Código

```cpp
void insertion_sort(vector< int > &nums) {
  for(int i = 1; i < nums.size(); i++) {
    int x = nums[i];
    int j = i;
    while(j > 0 && nums[j - 1] > x) {
      nums[j] = nums[j - 1];
      j--;
    }
    nums[j] = x;
  }
}
```

¿Cuál sería la complejidad?
$O(n^2)$

Part II

SORT DE C++

Sort de C++

- La librería estándar de C++ viene con su propio Sort
- Este usa un algoritmo llamado introspection sort, que tiene complejidad $O(n \log n)$
- Puede comparar cualquier cosa que tenga función de comparación

```
1 sort(comienzo, final);
2 sort(comienzo, final, funci n comparaci n);
```

Uso en arreglos

```cpp
int main() {
  int n;
  cin >> n;
  int arr[n];
  for(int i = 0; i < n; i++) {
    cin >> arr[i];
  }

  sort(arr, arr + n);

  for(int i = 0; i < n; i++) {
    cout << arr[i] << " ";
  }
  cout << "\n";
  return 0;
}
```

Uso en vectores

```cpp
int main() {
  int n;
  cin >> n;
  vector< int > arr(n);
  for(int i = 0; i < n; i++) {
    cin >> arr[i];
  }

  sort(arr.begin(), arr.end());

  for(int i = 0; i < n; i++) {
    cout << arr[i] << " ";
  }
  cout << "\n";
  return 0;
}
```

Función de comparación propia

```cpp
bool fun(pair< int, int > &a, pair< int, int > &b) {
  if(a.first == b.first) {
    return a.second < b.second;
  }
  return a.first > b.first;
}

int main() {
  int n;
  cin >> n;
  vector< pair< int, int > > arr(n);
  for(int i = 0; i < n; i++) {
    cin >> arr[i].first;
    cin >> arr[i].second;
  }

  sort(arr.begin(), arr.end(), fun);

  for(int i = 0; i < n; i++) {
    cout << "{" << arr[i].first << " " << arr[i].second << "} ";
  }
  cout << "\n";
  return 0;
}
```

REFERENCES I