Cheatography

Algoritmica Cheat Sheet

by greatcraic via cheatography.com/21020/cs/3866/

Algoritmi di ordinamento

ALGORITMO	Tempo	Spazio
Selection Sort	$\Theta(n^2)$	sul posto $\Theta(1)$
Insertion Sort *	Ottimo: Θ (n) Medio: $\Theta(n^2)$ Pessimo $\Theta(n^2)$	sul posto $\Theta(1)$
MergeSort *	Θ (n log n)	Θ(n)
QuickSort	Ottimo Θ(nlogn) Medio: Θ(nlogn) Pessimo: $\Theta(n^2)$	sul posto $\Theta(1)$ (ma spazio non costante x ricorsione)
HeapSort	Θ (n log n)	sul posto $\Theta(1)$
CountingSort *	Θ (n + k)	
RadixSort *	Θ (d(n+k))	

SelectionSort

```
for i =1 to n-1 {
  minimo = i;
  for j=i+1 to n {
    if (a[j] < a[minimo]) minimo = j;
  }
  scambia a[j] e a[minimo];
}
```

Insertion Sort

```
for j = 2 to n {
  key = a[j];
  i = j - 1;
  while (i > 0 && a[i] > key) {
     a[i+1] = a[i];
     i--;
  }
a[i+1] = key;
}
```

MergeSort

```
MergeSort(a, sx, dx)
if (sx < dx) {
  cx = (sx+dx)/2; //parte intera inferiore
  MergeSort(a, sx, cx);
  MergeSort(a, cx+1, dx);
  Merge(a, sx, cx, dx);
}
Merge(a, sx, cx, dx)
n1 = cx - sx + 1;
n2 = dx - cx;
L = nuovo array di dim n1+1
R = nuovo array di dim n2+1
for(i = 1; i ≤ n1; i++) L[i] = a[sx+i-1];
for(j = 1; j ≤ n1; j++) R[j] = a[cx+j];
L[n1+1] = ∞ //sentinella
R[n2+1] = ∞ //sentinella
i = 1;
j = 1;
for(k=sx; k ≤ dx; k++) {
  if (L[i] ≤ r[j]) {a[k] = L[i]; i++;}
  else {a[k] = R[j]; j++}
]
```

QuickSort

```
QuickSort(A, p, r)
if (p < r) {
  q = partition(A, p, r);
  QuickSort(A, p, q-1);
  QuickSort(A, q+1, r);
}
Partition(A, p, r) {
//nella versione random generare i = numero
//random tra p e r
//e metterlo in fondo
  x = A[r];
  i = p-1;
  for j = p to r -1 {
    if(a[j] ≤ x) {
      i++;
      scambia A[i] con A[j];
    }
  }
scambia(A[i+1], A[r]);
return i+1;
```

Heap

```
Max-Heapify(A, i)
 l = Left(i);
 r = Right(i);
 if l ≤ A.heapsize and A[l] > A[i]
   massimo = l;
 else massimo = i;
 if r ≤ A.heapsize and A[r] > A[massimo]
   massimo = r;
 if massimo ≠ i
   scambia A[i] con A[massimo];
   Max-Heapify(A, massimo);
Build-Max-Heap
 A.heapsize = A.length;
 for i = A.length/2 downto 1
   Max-Heapify(A, i);
Heapsort(A)
 Build-Max-Heap(A);
 for i = A.length downto 2
   scambia A[1] con A[i]
   A.heapsize--;
   Max-Heapify(A, 1);
```

Code di priorità

```
Heap-Maximum(A) return A[1];
Heap-Extract-Max(A)
 if A.heap-size < 1
  error "underflow dell'heap"
 max = A[1];
 A[1] = A[A.heapsize]
 A.heapsize--;
 Max-Heapify(A, 1);
 return max;
Heap-Increase-Key (A, i, key)
 if key < A[i]
  error "la nuova chiave è più piccola di
        quella corrente";
 A[i] = key;
 while i > 1 and A[Parent(i)] < A[i]
  scambia A[i] con A[Parent(i)];
  i = Parent(i);
Max-Heap-Insert(A, key)
 A.heapsize++;
 A[A.heapsize] = -∞;
 Heap-Increase-key (A, A.heapsize, key)
```

By **greatcraic**
cheatography.com/greatcraic/

Not published yet.
Last updated 12th April, 2015.

CountingSort

CountingSort (A, k)
C[0..k] nuovo array di dimensione k
//nota che qui conto da 0
for i = 0 to k
  C[i] = 0;
for i = 0 to n
  C[A[i]]++;
for i = 1 to k
  C[i] = C[i-1] + C[i];
for i = n downto 1 {
  B[C[A[i]]] = A[i];
  C[A[i]]--;
}

RadixSort

RadixSort(A, d)
for i = 1 to d
  usa un ordinamento stabile per ordinare
  l'array A sulla cifra i

Ricerca Binaria (tempo log n)

RicercaBinaria(a, sx, dx, k)
 if (sx > dx) return -1;
 cx = sx + dx / 2;
 if (a[cx] == k) return cx;
 if (a[cx] > k)
  return RicercaBinaria(a, sx, cx-1, k);
 else
  return RicercaBinaria(a, cx+1, dx, k);
RicercaBinariaSx(a, sx, dx, k)
 if (sx > dx) return -1;
 if (sx == dx) {
  if (a[sx] == k) return sx;
  else return -1;
 }
 cx = sx + dx / 2;
 if (a[cx] ≥ k)
  return RicercaBinaria(a, sx, cx, k);
 else
  return RicercaBinaria(a, cx+1, dx, k);

Randomized-Select (QuickSelect)

//Trova la i-esima occorrenza + piccola in
//tempo atteso lineare
Randomized-Select(A, p, r, i)
 if (p == r)
  return A[p];
 q = Randomized-Partition(A, p, r); //vedi
                    //quicksort
 k = q - p + 1;
 if (i == k) // il valore del pivot è la
        // soluzione
  return A[q];
 else if (i < k)
  return Randomized-Select(a, p, q-1, i)
 else
  return Randomized-Select(a, q+1, r, i-k)