

Algoritmi di ordinamento e selezione

- Selection Sort
 - trova il minimo elemento tra gli n iniziali e lo scambia con quello che occupa la prima posizione; ripete il procedimento sui restanti $n - 1$ elementi, poi sui rimanenti $n - 2$, e così via fino ad aver esaurito gli elementi
 - complessità
 - * $O(n^2)$
- Insertion Sort
 - considera gli elementi uno per volta consecutivamente: ogni volta che viene considerato un nuovo elemento, lo inserisce nella posizione giusta rispetto agli altri già considerati e ordinati, traslando di una posizione verso destra tutti gli elementi maggiori
 - complessità
 - * $O(n^2)$ vettore ordinato alla rovescia
 - * $O(n)$ vettore ordinato
- Merge Sort
 - divide il vettore in due sequenze di $n/2$ elementi ciascuno, ordina ciascuna sequenza, e poi fonde le due metà ordinate in un'unica sequenza ordinata
 - complessità
 - * $O(n \log n)$
- Quick Sort
 - seleziona un elemento del vettore, detto perno, attorno al quale riarrangiare gli altri elementi. Tutti gli elementi più piccoli del perno vengono spostati in posizioni del vettore che precedono quella del perno, mentre gli elementi più grandi del perno sono spostati in posizioni che la seguono. Lo stesso algoritmo è poi riapplicato alle due porzioni di elementi minori e maggiori del perno.
 - complessità
 - * $O(n^2)$ vettore ordinato
 - * $O(n \log n)$ caso medio
- Counting (Pigeonhole) Sort
 - scandisce inizialmente il vettore A e, avvalendosi di un vettore di appoggio B , registra in $B[A[j]]$ il numero di occorrenze del valore $A[j]$, per $j = 1, 2, \dots, n$. Successivamente, la procedura scandisce il vettore B , per $i = 1, 2, \dots, k$ e scrive il valore i nel vettore A per $B[i]$ volte
 - complessità

- * $O(n + k)$
- * $O(n)$ se k è $O(n)$

- Selezione in tempo atteso lineare

Alberi

- definizioni
 - altezza
 - radice, nodo interno, foglia
- implementazione
 - alberi binari
 - array figli
 - vettore dei padri
 - primo figlio / fratello
- visita in-ordine
 - prima figli sinistri, poi radice, poi figli destri
- visita pre-ordine
 - prima radice e poi nell'ordine i figli
- visita post-ordine
 - prima i figli nell'ordine e poi la radice
- visita ampiezza
 - vengono visitati i nodi per livello

Strutture dati - 2

Tabelle hash

- Tabelle ad indirizzamento diretto
 - array grande quanto il dominio delle chiavi
- Funzioni hash: metodo della divisione
 - $H(k) = k \bmod m$
- Funzioni hash: metodo della moltiplicazione

$$- H(k) = m((kA) \bmod 1) \quad 0 < A < 1$$

- Concatenamento

- Indirizzamento aperto

- Ispezione lineare

$$- H(k, i) = (H(k) + h \cdot i) \bmod m$$

- Ispezione quadratica

$$- H(k, i) = (H(k) + h \cdot i^2) \bmod m$$

- Doppio hashing

$$- H(k, i) = (H(k) + i \cdot H'(k)) \bmod m$$

Grafi - algoritmi vari

Minimum spanning tree

- Kruskal

– analizza gli archi in ordine crescente di peso; un generico arco è aggiunto all'albero T se non forma circuiti con gli archi inseriti in T

– complessità

$$* O(m \log n)$$

- Prim

– parte da un albero T contenente uno specifico nodo radice r , e accresce T aggiungendo sempre l'arco con peso minore fra quelli che "escono" da T , ovvero fra quelli che uniscono un nodo T con un nodo non incluso in T

– complessità

$$* O(m \log n)$$

Cammini minimi, singola sorgente

- Bellman-Ford

– coda

– complessità

$$* O(n(n + m)) = O(nm)$$

- Cammini minimi su DAG

- Dijkstra
 - coda con priorità
 - complessità
 - * $O(n^2)$

Cammini minimi, sorgenti multiple

- Floyd-Warshall
- Johnson
 - coda con priorità realizzata con un heap binario
 - complessità
 - * $O(m \log n)$
 - * $O(n \log n)$ per grafi sparsi, nei quali il numero m di archi è $O(n)$
 - * $O(n^2 \log n)$ per grafi densi, dove m è $\Omega(n^2)$

Problemi di flusso

- Ford-Fulkerson
 - richiede di trovare un cammino aumentante qualsiasi, utilizzando un qualunque algoritmo di visita. Assumendo che le capacità siano rappresentate da valori interi, ogni camminaumentante incrementa il flusso di almeno 1
 - complessità
 - * $O((n + m)|f^*|)$
- Edmonds-Karp
 - la ricerca del cammino aumentante è effettuata tramite una visita in ampiezza
 - complessità
 - * $O(nm^2)$