**DOMANDA A RISPOSTA APERTA- ESAME 23/1/2023**

Descrivere il concetto di costo dei programmi asintotico nel caso peggiore, la notazione O-grande e il suo utilizzo per confrontare il costo di due diversi programmi. Descrivere l'algoritmo di ordinamento merge-sort e discutere il suo costo asintotico. Fornire un esempio di esecuzione dell'algoritmo merge-sort, mostrando tutti i passi eseguiti sul seguente vettore dì input:

3 5 1 8 2 9 6 4

**RISPOSTA:**

Il costo di un programma è un indicatore molto utile utilizzato soprattutto per valutare l’efficienza di un programma o confrontare due programmi. È un indicatore che si basa, principalmente, su spazio di memoria occupato dal programma e tempo di esecuzione. Quando si studia il costo di un programma si prendono in considerazione tre situazioni: caso migliore, caso medio e caso peggiore, dove quest’ultimo indica il tempo massimo impiegato dal programma di dimensioni n. Per avere una stima quanto più reale possibile si prende in considerazione il caso peggiore e si parla di costo asintotico, ovvero una stima che cresce all’aumentare della dimensione dell’input, trascura costanti moltiplicative o termini di ordine inferiore rendendo i calcoli più facili. Per fare ciò, si utilizza la notazione matematica O grande, che ci dice che date due funzioni, da un certo valore n0 fissato, la dimensione della prima non supera mai la seconda. Applicandolo al costo dei programmi, un programma ha costo O(n) se, nel caso peggiore, il programma esegue un numero di operazioni che è O(n).

Una pratica applicazione del costo di un programma è per gli algoritmi di ordinamento. Uno tra questi è il merge sort, che si basa sul principio dividi ed impera e il suo algoritmo in pseudocodice può essere così riassunto:

Sia data una sequenza di numeri di dimensione n;

If(n<2) allora la sequenza è già ordinata;

Else ordina la metà sinistra;

Ordina la metà destra;  
fondi le due parti ordinate in un array ordinato.

Vediamo un esempio con l’array in input: 3 5 1 8 2 9 6 4

Si parte con le divisioni ricorsive fin quando la dimensione è strettamente minore di 2.

Dunque, si ha 3 5 1 8 2 9 6 4;

Poi ancora 3 5 1 8;

Poi ancora 3 5 1 8

A questo punto le divisioni si fermano e partono le fusioni in modo da ordinare le varie parti. Precisamente si ha 3 5 e 1 8 (che erano già ordinati).

E si passa a fondere al livello più esterno ottenendo 1 3 5 8.

Lo stesso procedimento lo si fa per la seconda metà:

Si divide ricorsivamente fino ad ottenere 2 9 6 4.

Si fondono ottenendo 2 9 e 4 6;

Si fonde al livello più esterno ottenendo 2 4 6 9.

A questo punto manca solo la fusione finale che ci restituisce l’array ordinato 1 2 3 4 5 6 8 9.

Studiamone ora il costo computazionale:

Notiamo che l’algoritmo si compone di una parte di divisione e una di fusione. Per quanto riguarda la parte di fusione, questa ha un costo computazione fisso che è O(n). Si osservi che invece le divisioni vengono effettuate a prescindere dalla configurazione dell’input (anche se l’array in input dovesse essere ordinato correttamente). Dunque, non esiste un caso peggiore o un caso migliore nel merge sort. Alla k esima divisione l’array avrà dimensioni n/2^k e tale procedimento continuerà fin quando k<=log n/2. Dunque, complessivamente il merge sort ha costo O (n log n).