

Informatica 1

Corso di Laurea Triennale in Matematica

Gianluca Rossi

`gianluca.rossi@uniroma2.it`

Dipartimento di Matematica
Università di Roma "Tor Vergata"

a-3: **Vettori dinamici**



Il problema

Inserire ed eliminare un elemento dal fondo di un array.

- Supponiamo che il vettore v abbia capacità c e contenga n elementi.
- Cancellazione: nessun problema, $n - 1$.
- Inserimento: se $n < c$ nessun problema, si inserisce in $v[n]$;
- Inserimento: se $n = c$ si deve creare un vettore più grande e copiare gli elementi dal vecchio vettore.
- Ogni inserimento può richiedere fino a n passi.



Distribuire il costo dovuto alla copia degli n elementi su n operazioni.

Inserimento di e

- Se $n < c$, $v[n++] = e$;
- Se $n = c$, creazione di un vettore di dimensione (capacità) $2c = 2n$ e copia degli n elementi nel nuovo vettore.

Cancellazione

- Se $n > c/4$, $n--$;
- Se $n = c/4$, creazione di un vettore di dimensione $c/2 = 2n$ e copia degli n elementi nel nuovo vettore.



- Subito dopo un raddoppio o dimezzamento di capacità $c = 2n$
- Prima del raddoppio della capacità del vettore occorrono **almeno** n inserimenti ognuno di costo $O(1)$. Costo $O(n)$.
- Prima del dimezzamento della capacità del vettore occorrono **almeno** $n/2$ cancellazione ognuno di costo $O(1)$. Costo $O(n/2)$.
- I costi $O(n)$ e $O(n/2)$ vanno “spalmati” su n e $n/2$ operazioni rispettivamente, ovvero $O(1)$ complessivo per ogni operazioni.



Struttura

```
struct vector{  
    float *array;  
    int cpty; /* capacita' */  
    int n; /* numero elementi (prima posizione libera) */  
};  
typedef struct vector vector;
```



Creazione

```
vector newvector(int c){  
    vector v;  
    v.array = (float*)malloc(sizeof(float)*c);  
    v.cpty = c;  
    v.n = 0;  
    return v;  
}
```



Inserimento

```
vector insert(vector v, float e){
    if(v.n<v.cpty){
        v.array[v.n]=e;
        v.n++;
        return v;
    } else { /* raddoppio la dimensione del vettore */
        vector new_v;
        int i;
        new_v = newvector(2*v.cpty);
        for(i=0; i<v.n; i++)
            new_v.array[i] = v.array[i];
        new_v.array[v.n]=e;
        new_v.cpty = 2*v.cpty;
        new_v.n = v.n+1;
        return new_v;
    }
}
```

Cancellazione

```
vector delete(vector v){
    if(v.n > (v.cpty)/4){
        v.n--;
        return v;
    } else { /* dimezzo la capacita' del vettore */
        vector new_v;
        int i;
        new_v = newvector((v.cpty)/2);
        for(i=0; i < v.n-1; i++)
            new_v.array[i] = v.array[i];
        new_v.cpty = (v.cpty)/2;
        new_v.n = v.n-1;
        return new_v;
    }
}
```