**STRUTTURE DATI**

* **Stazioni**: Albero binario di ricerca; i campi saranno:
  + Km 🡪 *int*; chilometri di distanza tra l’inizio della strada e la stazione considerata;
  + Dx 🡪 *puntatore* al figlio destro;
  + Sx 🡪 *puntatore* al figlio sinistro;
  + Auto 🡪 *puntatore* al vettore delle auto disponibili alla stazione considerata;
  + MaxAuto 🡪 *int*; maggiore autonomia tra le auto della stazione;
  + AutoIndex 🡪 int; prima posizione libera nel vettore delle auto;
  + Aux 🡪 *int;* attributo utile per gli algoritmi di ricerca del percorso ottimale.
* **Automobili**: vettore di interi:
  + Autonomia 🡪 chilometri percorribili dall’auto considerata.

**ALGORITMI**

**Aggiungi stazione**

Dato un chilometro, un numero di auto e la loro autonomia, inserisce la stazione nella posizione corretta all’interno dell’autostrada:

1. Si inserisce la sola stazione al posto giusto nell’albero:
   1. Se non ci sono altre stazioni, la si inserisce come radice;
   2. Altrimenti, si scorrono le stazioni alla ricerca del punto di inserimento:
      1. Se si trova una stazione con lo stesso chilometro, si termina comunicando l’impossibilità di aggiungere la nuova stazione;
      2. Se si trova una stazione con chilometro maggiore, si controlla nel sottoalbero sinistro di tale stazione, altrimenti nel destro;
      3. Si inserisce quindi la stazione nella foglia opportuna.
2. Si salva il numero di auto in una variabile N;
3. Si ripete N volte l’inserimento di un’auto nella stazione appena posizionata;
4. Al termine di tutto, si stampa l’esito richiesto.

**Demolisci stazione**

Dato un chilometro, rimuove la stazione relativa e le corrispondenti automobili:

1. Si scorre l’albero alla ricerca della stazione indicata:
   1. Se non la si trova dopo aver controllato tutto, terminiamo con output “non demolita”;
   2. Se la troviamo, effettuiamo la rimozione da albero binario di ricerca, distinguendo i tre casi possibili (no figli, un figlio, due figli)
2. Se arriviamo in fondo dopo aver effettuato la cancellazione, terminiamo e stampiamo “demolita”.

**Aggiungi auto**

Data una stazione ed una autonomia, si inserisce in quella stazione un’auto con tale autonomia:

1. Si cerca la stazione di input nell’albero delle stazioni:
   1. Se non la si trova, si restituisce 0 al chiamante;
   2. Se la si trova, si tiene da parte un puntatore ad essa e si prosegue;
2. Trovata la stazione, si confronta l’autonomia in input con la massima autonomia della stazione;
   1. Se è maggiore, si deve annotare il nuovo max;
3. A questo punto, si inserisce l’auto di input nella prima posizione libera del vettore della stazione trovata e si ritorna il valore 1;
4. Ritornati al chiamante, se si riceve 1 si stampa “aggiunta”, sennò si stampa “non aggiunta”.

**Rottama auto**

Data una certa stazione e l’autonomia di un’auto, rimuove tale auto dalla stazione:

1. Si cerca la stazione di input nell’albero:
   1. Se non la si trova, si restituisce 0 al chiamante;
   2. Se la si trova, si tiene da parte un puntatore ad essa e si prosegue;
2. Trovata la stazione, si confronta l’autonomia in input con la massima autonomia della stazione;
   1. Se è minore non si fa nulla, se è maggiore si ritorna subito 0 (non c’è auto maggiore nel vet);
   2. Se è uguale, si setta un flag a 1 (dovremo lanciare la ricerca del max dopo la cancellazione);
3. Trovata la stazione, si cerca nel suo vettore di auto l’autonomia di input:
   1. Se non la si trova, si restituisce 0 al chiamante;
   2. Se la si trova, si spostano tutti i valori dopo sovrascrivendo il valore da cancellare, si aggiorna il max della stazione (se serve) e si ritorna 1;
4. Ritornati al chiamante, se si riceve 1 si stampa “demolita”, sennò si stampa “non demolita”.

**Trovapercorso: Sx – Dx**

Con la prima passata si compila per ogni stazione il campo aux, che indichi la stazione più a sinistra che può raggiungerla:

1. Si inizializza un puntatore Curr alla prima stazione, e un puntatore X al succ della prima stazione;
2. Si controlla se Curr può raggiungere X:
   1. Se sì, si scrive il km della stazione Curr nel campo apposito della stazione X;
   2. Se no, si tiene fermo X e si manda avanti Curr;
3. Si ripete finché abbiamo compilato anche il campo del nodo destinazione (in tal caso tutto ok);
4. Se Curr e X si sovrappongono, si termina stampando “nessun percorso”.

A questo punto, se siamo nel caso ok, si possono sfruttare le informazioni ricavate precedentemente per trovare e stampare il percorso cercato:

1. Si imposta il nodo destinazione come “corrente”, fuori dal ciclo;
2. Si aggiunge il km corrente ad una pila di interi (si può usare la pila di sistema? Funzione ricorsiva);
3. Il nuovo corrente diventa il nodo indicato nel campo compilato prima;
4. Si ripete finché non si è aggiunto il nodo sorgente alla pila;
5. Terminato ciò, si stampa la pila.

**Trovapercorso: Dx – Sx**

Prima di tutto si devono compilare i campi aux delle stazioni, associando ad ogni nodo X la distanza minima dalla sorgente a X:

1. Inizializzo a 0 la distanza di S da S;
2. Il nodo S diventa Curr: considero il suo pred X e controllo se C può raggiungerlo;
   1. Se sì, imposto la distanza S->X del nodo X a 1 + distanza S->C (al passo 1 si fa 1+0);  
      mando avanti il puntatore pred al pred di X e ripeto;
   2. Se no, mando avanti il puntatore di C al nodo pred di C;
3. Ripeto finché non arrivo alla destinazione (ok) oppure finché i due puntatori si sovrappongono (in tal caso termino dicendo che non esiste percorso);

Fatto ciò, si deve procedere con l’effettiva ricerca del percorso più breve e più allineato a sinistra possibile:

1. Con un puntatore Curr tengo traccia del percorso ideale che sto tracciando: inizio con Curr = Dest;
2. \*Finché la sorgente non può raggiungere il Curr direttamente (campo settato == 1), ripeto:
   1. Aggiungo il km di Curr ad una pila;
   2. Imposto un puntatore X al succ di Curr, per scorrere tutte le stazioni;
   3. Imposto la minima distanza alla sorgente a +∞;
   4. Se X può raggiungere Curr, confronto il suo campo con la minima distanza corrente:
      1. Se è minore,   
         se X è adiacente alla sorgente aggiungo X e la sorgente alla pila e vado a stamparla,  
         aggiorno la minima distanza,  
         imposto il puntatore prox\_curr a X;
   5. X diventa il succ del precedente X, e continuo finché X non è il nodo sorgente;
   6. A questo punto, aggiorno il Curr con il prox\_curr trovato e ripeto tutto.

\*In realtà se siamo al secondo sotto algoritmo, vuol dire che almeno un percorso possibile esiste (ciò è garantito dalla buona riuscita del primo sotto algoritmo). Dunque, a un certo punto sicuramente si verificherà l’evento evidenziato, ed è lì che usciremo dall’algoritmo. Perciò, la condizione del passo 2. non serve, si può anche ripetere indefinitamente senza testare alcuna condizione.

**LIBRERIE**

Si possono utilizzare tutte le librerie standard del C:

* <assert.h> // debug aggressivo
* <ctype.h> // classificazione caratteri
* <errno.h> // costanti per errori numerici
* <float.h> // costanti per numeri float
* <limits.h> // costanti di limite
* <math.h> // funzioni matematiche
* <stddef.h> // include la macro NULL
* <stdio.h> // funzioni base di i/o
* <stdlib.h> // funzioni e costanti utili
* <string.h> // funzioni sulle stringhe
* <time.h> // manipolazione del tempo