Esercizio 9.1 Giovanni Pizzenti s249066

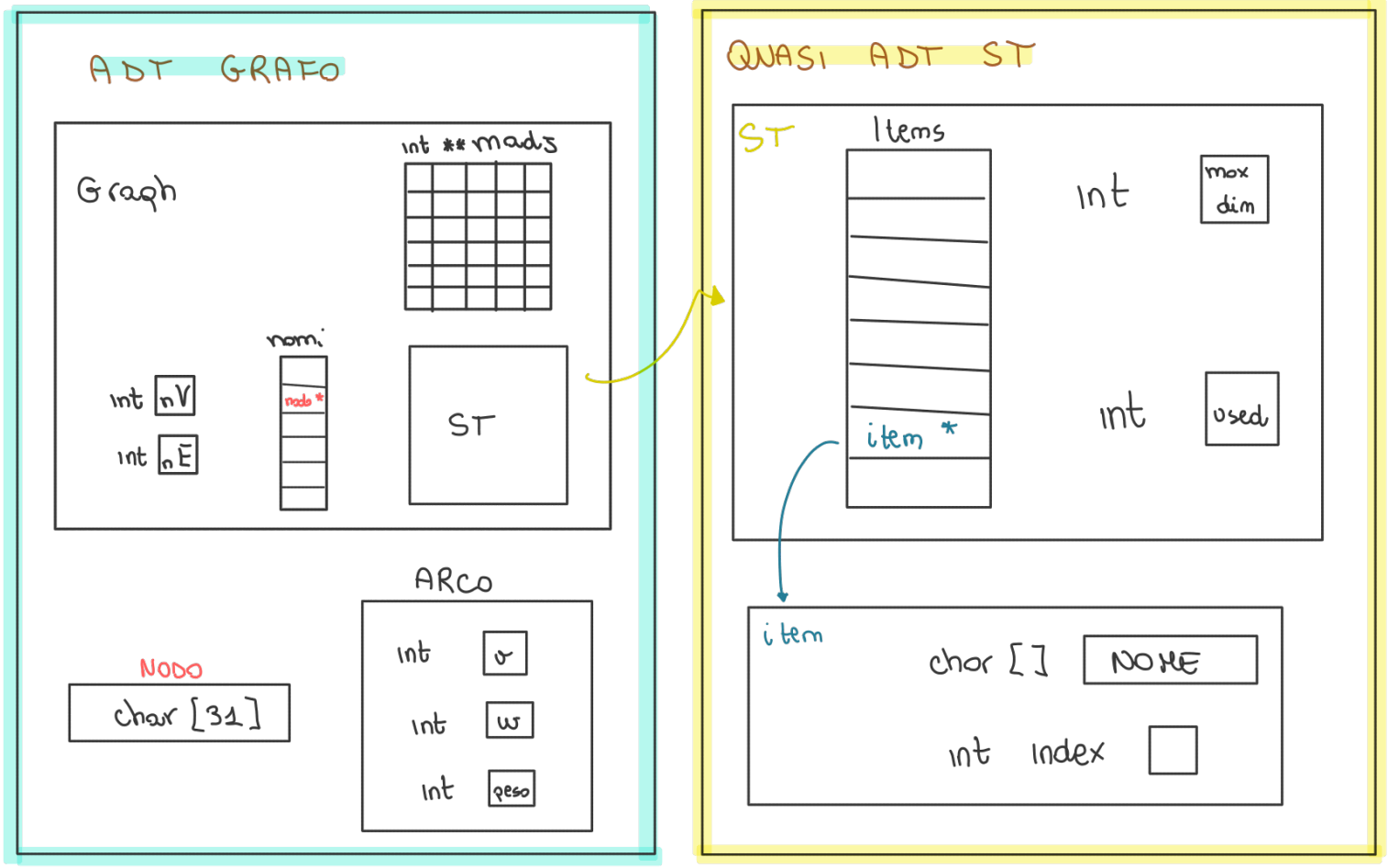
Strutture dati

Per la tabella di simboli ho deciso di utilizzare un quasi ADT le cui strutture dati sono:

- Il tipo item che consiste in una struct con una stringa (nome) e un intero (indice);
- Il tipo tabella di simboli che consiste in una struct con un vettore di Item, la dimensione allocata e la dimensione effettivamente utilizzata;

Per il grafo ho implementato un ADT di prima classe le cui strutture dati sono:

- Il tipo arco che comprende una struct con tre interi: l'indice del nodo di partenza, del nodo di arrivo e il peso;
- Il tipo grafo che comprende: la matrice delle adiacenze, la tabella di simboli, un vettore di stringhe con i nomi dei vertici, il numero di archi e il numero di vertici;



Algoritmi

I primi due punti dell'esercizio richiedono di trovare tutti gli insiemi di archi a cardinalità minima la cui rimozione renda il grafo di partenza un DAG e tra questi salvare quelli a peso massimo. Ho iniziato facendo una DFS preliminare in modo tale da non iterare su quei grafi che sono già dei DAG; infatti, con la DFS, mi basta trovare un arco back per poter dire che il grafo ha un ciclo e non è quindi un DAG. Nel caso in cui il grafo sia invece ciclico utilizzo il powerset con il modello delle combinazioni semplici, il quale, partendo da cardinalità 1, considera tutti i possibili set di archi che se rimossi dal grafo iniziale lo trasformano in un DAG, quindi salvo quelli a peso massimo così da ottenere il DAG a peso minimo.

L'ultimo punto richiede di trovare i cammini massimi per tutti i nodi sorgente del DAG trovato sopra. Inizio ordinando topologicamente il DAG usando la funzione DAGrts che a sua volta chiama la TSdsfR (entrambe presenti nelle slide capitolo 12), in seguito chiamo la funzione DAGmaxDis che riceve come parametro il grafo già ordinato e applica la relaxation inversa, cioè testa se la distanza dal vertice di partenza al vertice di arrivo, salvata nel vettore dist[] e aggiornata volta per volta, è minore della distanza che si ottiene partendo sempre dallo stesso vertice ma arrivando ad un altro vertice intermedio, sommata al peso dell'arco tra il vertice intermedio e il vertice finale. Se questa condizione è verificata, devo aggiornare il vettore dist[] con il nuovo valore dato dalla somma di d[u] + w[u,v]. Questo procedimento viene applicato a tutti i nodi intermedi raggiungibili, utilizzando un ciclo for.