Spiegare la realizzazione di grafi mediante matrice di adiacenza e matrice di incidenza, fornendo vantaggi e svantaggi di ognuna:COS'E' UNA MATRICE DI INCIDENZA?Le più semplici rappresentazioni di grafi utilizzano matrici. Tra queste le più usate sono la matrice di incidenza nodi-archi e la matrice di adiacenza nodi-nodi.La matrice di incidenza nodi-archi è una matrice rettangolare B ,= [bik] di dimensioni n x m, dove ciascuna riga rappresenta un nodo e ciascuna colonna rappresenta un arco, tale che:

- l, se l'arco k-esimo esce dal nodo i,bik = +1, se l’arco k-esimo entra nel nodo i, 0, altrimenti.Se il grafo è non orientato, invece, si ha:

bik = 1, se l'arco k-esimo è incidente nel nodo i

0, altrimenti.

VANTAGGI E SVANTAGGI MATRICE INCIDENZA:La matrice di incidenza è molto utile per rappresentare un grafo in certi problemi di PROGRAMMAZIONE LINEARE, nei quali si cerca un vettore x di m incognite (reali o intere) associate agli archi che soddisfi un sistema di equazioni del tipo Bx = c, dove c è un vettore di n termini noti interi. Purtroppo, con questa rappresentazione,lo spazio di memoria occupato è sempre θ(nm). Inoltre, dato un nodo, non è agevole ricavare l’insieme di adiacenza. Infatti per calcolare un qualsiasi A(u) occorre scandire la riga u di B alla ricerca delle colonne k con elementi buk = -1 e per ciascuna colonna k scandirla per individuare l'indice di riga v tale che bvk = + 1, per un totale di θ(nm) tempo. COS'E` UNA MATRICE DI ADIACENZA?La matrice di adiacenza nodi-nodi, invece, è una matrice quadrata E=[ eij ] di dimensioni n x n, tale che: 1, se (i,j)∈A,

eij = 0, se (i,j)∉A.Se il grafo è non orientato, allora tale matrice è simmetrica, ovvero eij = eji per ogni i e j.

VANTAGGI E SVANTAGGI MATRICE ADIACENZA;Con questa realizzazione è banale verificare in tempo O(1) se un dato arco e presente nel grafo oppure no. Purtroppo, il tempo per ricavare l’insieme di adiacenza A(u) di un qualsiasi nodo u è sempre θ(n), indipendentemente dalla cardinalità di A(u), perché occorre scandire un'intera riga della matrice alla ricerca degli elementi buv = 1. Inoltre, lo spazio di memoria occupato è θ(n2) e il tempo per esaminare tutti gli archi è θ(n2) anche se il grafo è sparso, cioè contiene un numero m di archi che è O(n). Se il grafo è pesato, allora nella matrice si utilizzano i pesi degli archial posto degli elementi binari. Se pij è il peso dell’ arco (1, 1), allora la matrice E diventa:

eij=  pij se (i,j) ∈A + ∞ (oppure - ∞ ) se (i,j)∉A

Strutture dati per la sua implementazionePer visitare" almeno una volta ogni nodo ed ogni arco di un grafo orientato fortemente connesso (o non orientato e connesso). Il metodo effettua la visita in tempo ottimo ® (n + m) a partire da un generico nodo r: ad esempio il Breadth-First-Search (BFS), ovvero visita in ampiezza (detta anche a ventaglio).Nella visita BFS i nodi sono visitati in ordine di distanza crescente dal nodo partenza r, dove la distanza da r ad un generico nodo u è il minimo numero di archi in un cammino (o catena) da r ad u. Tale metodo di visita è una estensione di una vista per livelli di un albero radicato, in cui i figli di un nodo sono visitati dopo aver visitato tutti gli altri nodi che stanno allo stesso livello del padre.In entrambi i metodi, occorre mantenere una lista dei nodi che sono già stati visitati. Poiché la DFS tornadal più recente vicolo cieco, tale lista è di fatto una pila. D’altro lato, poiché la BFS visita i nodi più vicini ad r prima di quelli più lontani, tale lista e una coda.La BFS è descritta utilizzando gli operatori delle code:Immagine che contiene testo

Descrizione generata automaticamente

La verifica di non appartenenza del nodo v alla coda può essere facilmente effettuata in tempo O(1) introducendo nella procedura BFS un opportuno vettore booleano in cui il bit v è impostato a vero quando if è inserito in Q ed è rimesso a falso quando v è estratto da Q. Questa verifica è necessaria per evitare che la coda contenga più di una copia di v, qualora v sia stato precedentemente inserito in coda durante l’esame di un arco (w, v) uscente da un nodo w già marcato visitato. Anche la "marcatura" di ciascun nodo può essere effettuata con un secondo vettore booleano. con ciascun bit u inizializzato a “falso”, che poi viene impostato a “vero” quando viene visitato il nodo u.Poiché ogni nodo del grafo è marcato una sola volta e. per ogni nodo v, viene scandito 1`intero insieme A(v) una sola volta, la complessità di entrambe le procedure è θ (n + m). Spiegare il concetto di collisione e le corrispondenti tecniche di gestione per dizionariLa realizzazione del dizionario con tabelle hash si basa sul concetto di ricavare direttamente dal valore della chiave la posizione che la chiave stessa dovrebbe occupare in un vettore. Con questa realizzazione, ciascuna operazione, pur avendo complessità O(n) nel caso pessimo, richiede tempo O(1) nel caso medio, Sia K l’insieme di tuttele possibili chiavi distinte e si memorizzi il dizionario in un vettore V di dimensione m. La soluzione ideale sarebbe quella di possedere una funzione d'accesso H: K 🡪{1,... ,m}che permetta di ricavare la posizione H(k) della chiave k nel vettore V in modo cheper ogni k1 ∈ K e k2 ∈ K, con k1 ≠ k2. risulti H(k1) ≠ H(k2). Per garantire tale biunivocità, ovviamente, m non può essere più piccolo di |K|

Per esempio, utilizzando un vettore con m : |K| posizioni, si potrebbe accedere direttamente alla posizione contenente la chiave in tempo O(l). Se |K| è grande, purtroppo. questo metodo è improponibile perché richiede uno spreco enorme di memoria.Per evitare inutili sprechi di memoria, la dimensione m del vettore va scelta in base al numero di chiavi attese, cioè che ci si aspetta possano essere effettivamente memorizzate contemporaneamente, e non in base a quello di tutte le chiavi possibili, la maggior parte delle quali potrebbe non apparire mai nel dizionario. Occorre quindi rinunciare alla biunivocità della funzione H ammettendo che chiavi diverse possano avere lo stesso indirizzo nel vettore, cioè che H(k1) = H(k2) anche se k1 ≠ k2.In tal caso. la soluzione estrema che minimizza l’uso della memoria sembra quella in cui c'è un unico indirizzo H(k) per ogni possibile chiave k, per esempio mantenendo tutte le chiavi in un'unica lista. In questo modo, la memoria richiesta è linearmente proporzionale al numero n di chiavi presenti nel dizionario, ma anche il tempo perle operazioni cresce con un inaccettabile O(n). Occorre pertanto una soluzione di compromesso, con m maggiore di 1 ma molto minore di |K|, che mantenga bassi sia il tempo di esecuzione delle operazioni (possibilmente O(1), come nel caso m = |K|).

Sia l’occupazione di memoria (possibilmente O(n) come nel caso m = 1).Rinunciando alla biunivocità di H, sorge il nuovo problema di scegliere le posizioni per memorizzare due chiavi k1e k2 che collidono, cioé con H( k1 = k2), qualora siano entrambe presenti contemporaneamente nel vettore. Quattro buoni metodi di generazione di indirizzi hash sono i seguenti. i primi tre presuppongono m = 2p per qualche p, mentre il quarto presuppone m dispari(meglio se primo):a) H(k)= int(b), dove b è un sottoinsieme di p bit di bin(k), solitamente estratti nelle posizioni centrali

b) H(k)= int(b), dove b è dato dalla somma modulo 2, effettuata bit a bit, di diversi sottoinsiemi di p bit di bin(k);

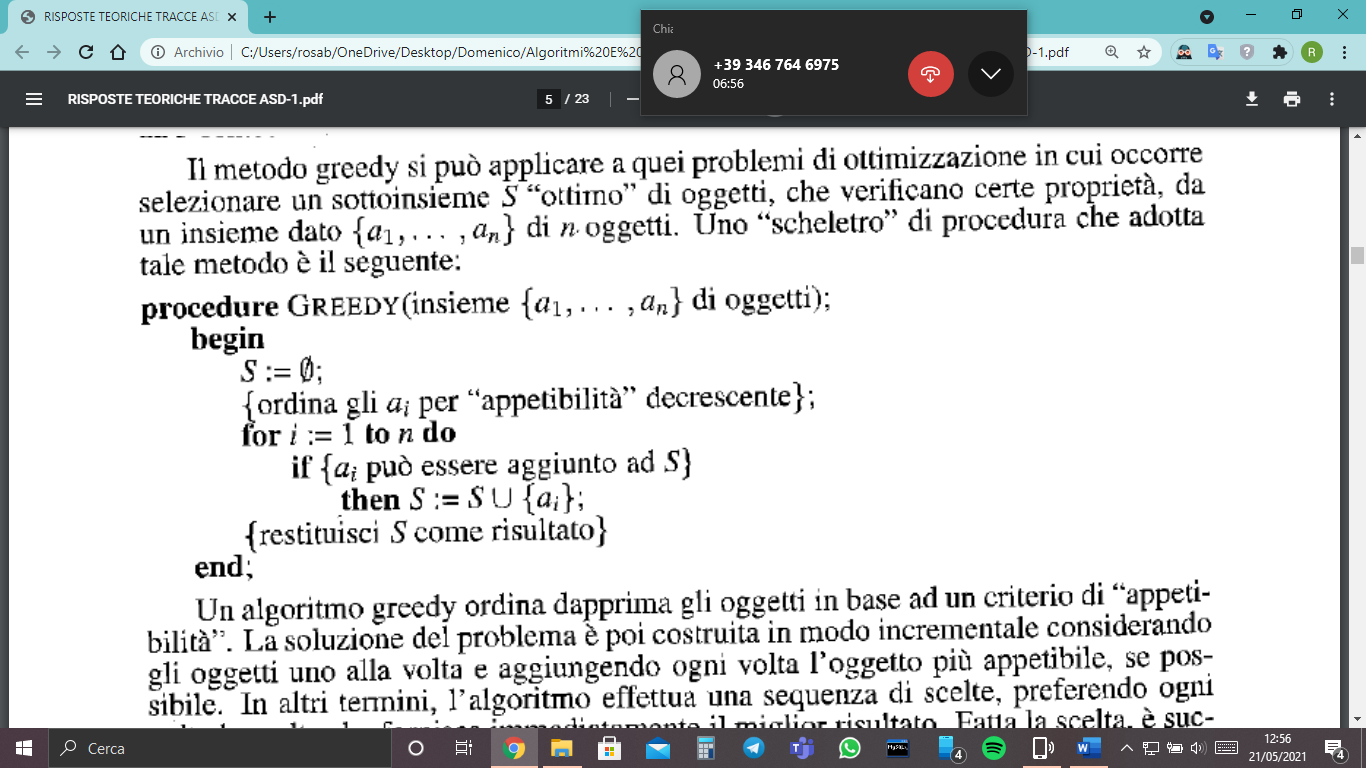
c) H(k)= int(b), dove b è un sottoinsieme di p bit estratti dalle posizioni centrali di bin(int(bin(k))2);d) H(k) è uguale al resto della divisione di int(bin(k)) per m.La funzione hash definita in (d) presuppone che m sia dispari, perché se m fosse uguale a 2p allora due numeri con gli stessi p bit finali darebbero sempre luogo ad una collisione. Questa funzione è la migliore dal punto di vista probabilistico e fornisce un eccellente distribuzione degli indirizzi H(k) nell’intervallo [0, m - 1], specie se m è un numero primo non troppo vicino ad una potenza di 2 (o a qualsiasi altra base con la quale il calcolatore utilizzato rappresenti i numeri). Se il calcolatore rappresenta numeri usando la base 2. allora alcuni valori adeguati per m sono: 13, 23, 47, 97, 193, 83, 769, 1531. 3067, 6143, 12289, 24571.Una soluzione vantaggiosa e l'uso di una tabella con liste di trabocco esempio:Immagine che contiene testo

Descrizione generata automaticamente

IN BREVE DIZIONARIO:Una collisione hash è una situazione che avviene quando due diversi input producono lo stesso output tramite una funzione hash. Con la funzione hash, si vuole ricavare l'output (un numero intero) che sia scorrelato dalla struttura dell'input (la chiave stessa). Da qui il termine “hash”, che in inglese significa “tritare" (e quindi “polpetta"): si sminuzza la chiave in pezzettini che sono rimescolati in modo da formare un intero che sia il più “casuale” possibile.

Ovviamente l'algoritmo che calcola la funzione hash non e casuale.Il dizionario può essere gestito con un array, in modo sequenziale, ma se la dimensione è molto grande è uno spreco di memoria. Bisognerebbe già sapere il numero di chiave attese nel dizionario. Se due chiavi k1 e k2 producono lo stesso output, h(k1) ==h(k2) con k1 ≠ k2, si ha una collisione. L'Hash Table comunemente più implementata per ridurre il numero di collisioni, è con lista di Trabocco. Le liste di trabocco contengono tutte chiavi che collidono in un'unica lista. (Eliminazione, cancellazione, aggiornamento) è sufficiente calcolare l’hash della chiave da ricercare, accedere alla lista di trabocco e scandirla fino a trovare la chiave desiderata.Fornire la specifica di problema di ricercaDefinizione: un problema di Ricerca p è un problema specificato con una quintupla del tipo: <L,S,R,S ∪ {⊥},qric>dove qric è la regola che definisce, in base ad R, la relazione Rqric contenente tutti e soli i seguenti elementi:

- Ogni coppia contenuta in R; Una coppia <I, ⊥> per ogni istanza i di p per la quale P non ha soluzioni.Spiegare la tecnica algoritmica greedyNella vita di tutti i giorni, spesso si adottano strategie che danno immediatamente buoni risultati, ma che a lungo andare si rivelano scadenti. È questa la strategia dell’ingordo (o “greedy”):Si supponga di avere monete da 11,5 e 1 centesimo, e di dover fare un resto di 15 centesimi. L’ingordo giudicherà la moneta dal più “appetibile” delle altre, e farà un resto con una moneta da 11 e quattro da 1, utilizzando cinque monete in tutto, mentre la soluzione ottima è data da tre monete da 5 centesimi!Il metodo greedy si può applicare a quei problemi di ottimizzazione in cui occorre selezionare un sottoinsieme S “ottimo” di oggetti, che verificano certe proprietà, da un insieme dato {al, . . . ,an} di n oggetti. Uno “scheletro” di procedura che adotta tale metodo è il seguente:

Un algoritmo greedy ordina dapprima gli oggetti in base ad un criterio di “appetibilità”. La soluzione del problema e poi costruita in modo incrementale considerando gli oggetti uno alla volta e aggiungendo ogni volta l’oggetto più appetibile, se possibile. ln altri termini, l’algoritmo effettua una sequenza di scelte, preferendo ogni volta la scelta che fornisce immediatamente il miglior risultato. Fatta la scelta, è successivamente risolto un sottoproblema dello stesso tipo ma di dimensione più piccola.

Fornire la specifica sintattica e semantica degli operatori cancnodo e cancarco per la struttura dati grafoSPECIFICA SINTATTICA CANCNODO & CANCARCO:CANCNODO: (nodo; grafo)🡪 grafoCANCARCO: (nodo, nodo, grafo)🡪 grafo

SPECIFICA SEMANTICA CANCNODO & CANCARCO:Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Fornire la specifica sintattica e semantica degli operatori insSottoAlbero e insPrimoSottoAlbero per la struttura dati Alberi n-ari

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Spiegare la realizzazione di alberi n-ari mediante vettore dei padri, liste di figli e con cursori, fornendo vantaggi e svantaggi di ognunaImmagine che contiene testo

Descrizione generata automaticamente

ALBERO N-ARIO TRAMITE LISTE DI FIGLI VANTAGGI E SVANTAGGI:una variante molto usata, poiché non si può associare ad ogni elemento un numero di puntatori uguale al massimo dei figli è la rappresentazione attraverso liste di figli che comprende:° il vettore dei nodi, in cui, oltre alle eventuali etichette dei nodi, si memorizza il riferimento iniziale di una lista associata ad ogni nodo;° una lista per ogni nodo, detta lista dei figli. La lista associata al generico nodo I contiene tanti elementi quanti sono i successori di I; ciascun elemento e il riferimento ad uno dei successori.per l'esempio precedente si avrebbe:Immagine che contiene testo

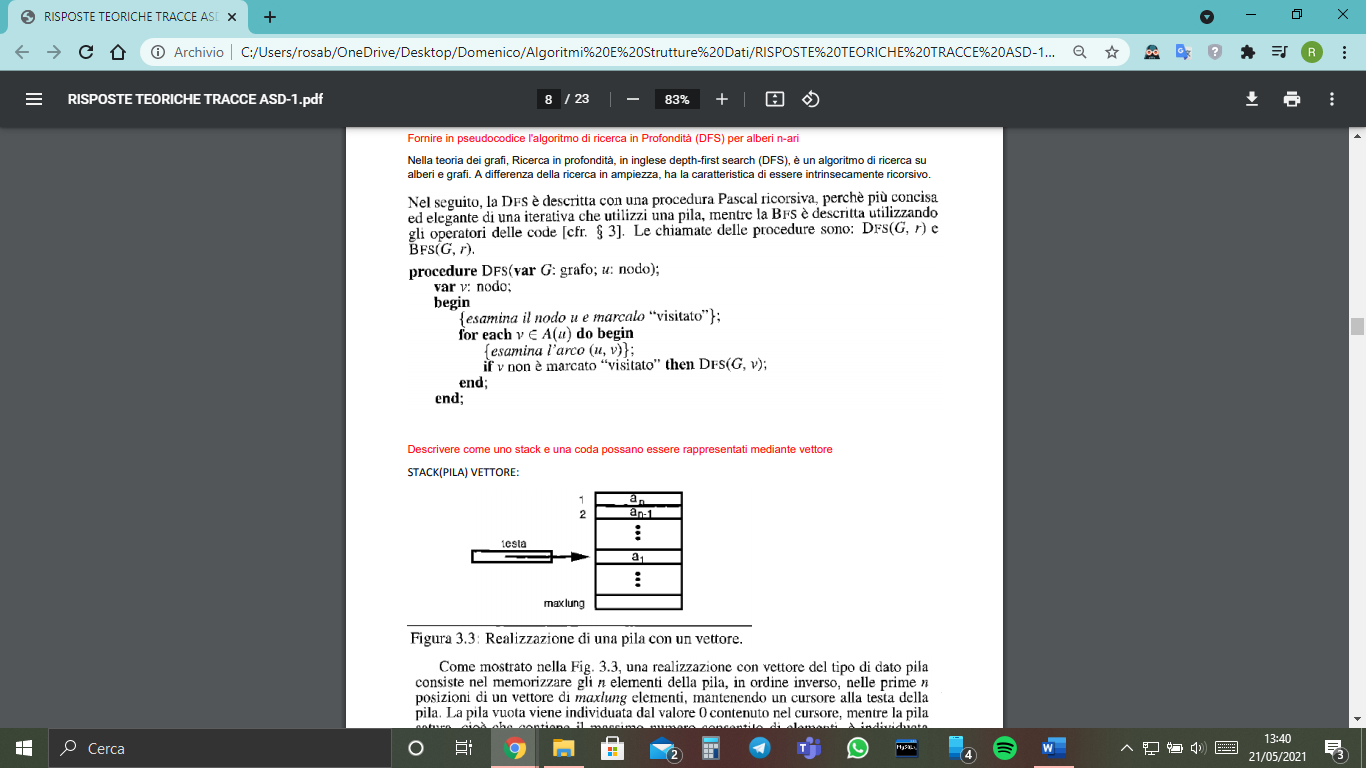
Descrizione generata automaticamente

Ovviamente, la realizzazione con liste dei figli presenta il vantaggio di poter scorrere velocemente tutti i figli di un dato nodo, ma non permette di realizzare efficientemente altri operatori come PADRE o SUCCFRATELLO. Infatti, per poter individuare il padre o il fratello successivo di un dato nodo u, occorre scandire nel caso pessimo tutte le n liste alla ricerca dell'unica lista nella quale compare il nodo u.RAPPRESENTAZIONE MEDIANTE LISTA PRIMOFIGLIO/FRATELLO

prevede la gestione di una lista e questo può essere fatto imponendo che tutti gli alberi condividano un'area comune (ad esempio un vettore nella realizzazione con cursori) e che ogni cella contenga esattamente due cursori: uno al primofiglio ed uno al fratello successivo. la realizzazione e simile a quella proposta per gli alberi binari con l'unica differenza che il cursore nel terzo campo punta al fratello. Naturalmente è possibile anche prevedere un cursore al genitore:Immagine che contiene testo

Descrizione generata automaticamenteFornire in pseudocodice l'algoritmo di ricerca in Profondità (DFS) per alberi n-ariNella teoria dei grafi, Ricerca in profondità, in inglese depth-first search (DFS), è un algoritmo di ricerca su alberi e grafi. A differenza della ricerca in ampiezza, ha la caratteristica di essere intrinsecamente ricorsivo.Nel seguito, la DFS è descritta con una procedura Pascal ricorsiva, perché più concisa ed elegante di una iterativa che utilizzi una pila. mentre la BFS è descritta utilizzando gli operatori delle code. Le chiamate delle procedure sono: DFS(G, r) e Best G, r).Immagine che contiene testo

Descrizione generata automaticamente

Descrivere come uno stack e una coda possano essere rappresentati mediante vettoreSTACK(PILA) VETTORE:.Come mostrato nella figura, una realizzazione con vettore del tipo di dato pila consiste nel memorizzare gli n elementi della pila, in ordine inverso, nelle prime n posizioni di un vettore di maxlung elementi. mantenendo un cursore alla testa della pila. La pila vuota viene individuata dal valore 0 contenuto nel cursore, mentre la pila satura, cioè che contiene il massimo numero consentito di elementi è individuata dal valore maxlung. FUORIPILA è realizzata decrementando il valore del cursore.mentre INPILA è realizzata incrementando il valore del cursore ed inserendo il nuovo elemento nella posizione del vettore individuata dal cursore.

Con questa realizzazione, ogni operazione della pila richiede tempo costante per essere eseguita. Per ottenere la stessa complessità anche per tutte le operazioni della coda, si utilizza un vettore “circolare”.CODA COME VETTORE CIRCOLARE:Si supponga di avere un array di maxlung elementi, indicati da 0 a maxlung -1, in cui l’elemento di indice 0 è considerato come successore di quello di indice maxlung - 1Immagine che contiene testo

Descrizione generata automaticamente

La coda è contenuta in n posizioni consecutive del vettore. per esempio secondo il verso orario, e i suoi elementi estremi sono individuati da due cursori. Per cancellare o per inserire un elemento, viene incrementato di uno (modulo maxlung) il rispettivo cursore di testa o di fondo.

In questo modo la coda "migra” lungo il vettore circolare e, ad ogni istante, si trova da qualche parte nel vettore, sempre in posizioni consecutive secondo il verso orario.Anziché utilizzare due cursori. per individuare la coda è sufficiente conoscere l’indice del primo elemento e la lunghezza della coda stessa, come mostrato nella figura.

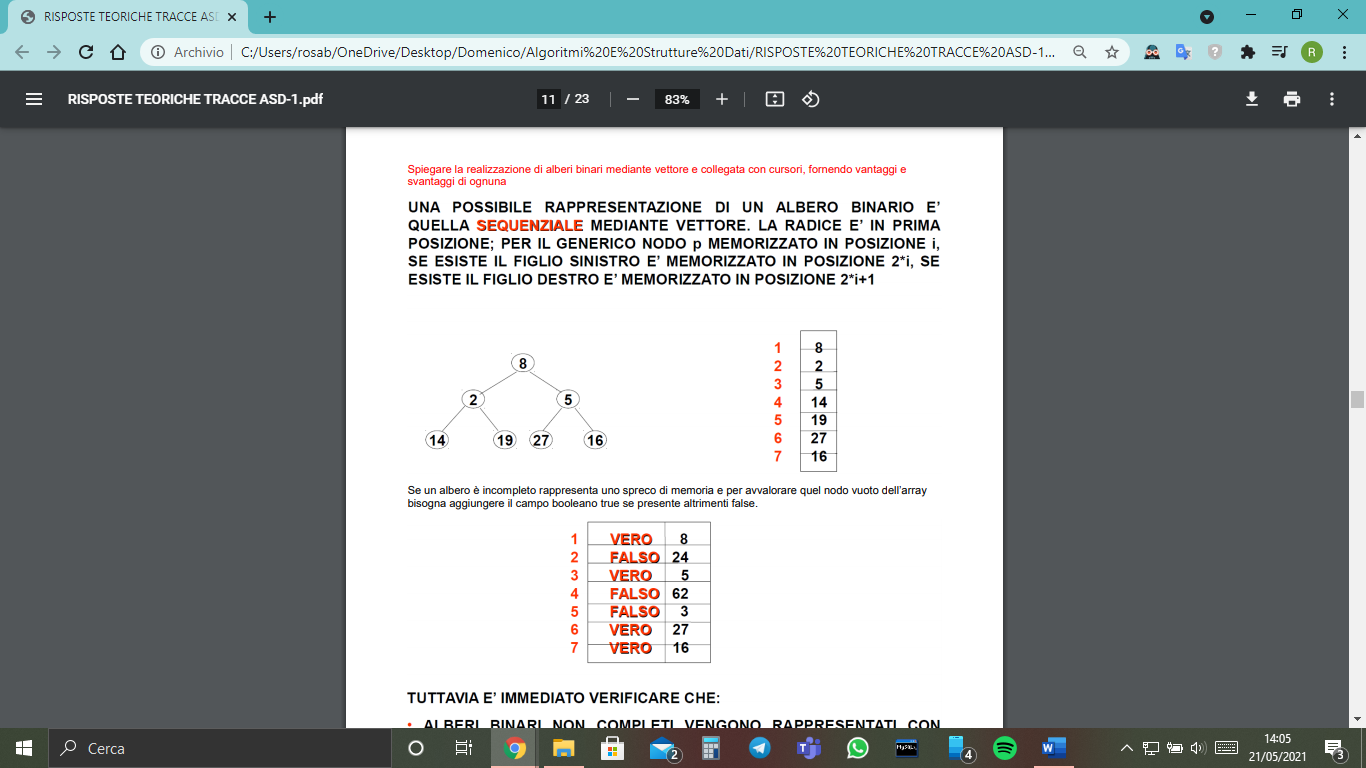
Infatti, se i è la posizione dell'elemento in testa alla coda. allora quella dell’ultimo elemento della coda è (i + n - 1) modulo maxlungQuesta tecnica, detta backtrack, è alla base di molti algoritmi di visita di alberi e grafi, ed è usata per generare sistematicamente tutte le soluzioni ammissibili di un problema (“to backtrack” significa "tornare sui propri passi").Certo, una tecnica non molto astuta, ma può rivelarsi molto utile.Motivare le differenze fra una tecnica greedy e una di backtracking quando applicate ad un problema di ottimizzazioneProblemi di ottimizzazione nei quali alle soluzioni ammissibili è associata una misura (o costo, o obiettivo) e si vuole trovare una soluzione ottima, cioè una soluzione ammissibile la cui misura sia minima o massima (ad esempio:trovare una foglia di un albero radicato il cui livello sia massimo).Quando si applica il backtracking, la tecnica greedy in un problema di ottimizzazione, stabilisce il tipo di output aspettato, sempre in base al quesito.Per esempio, l’algoritmo greedy è utile per ottenere una delle possibili soluzioni nel tempo più minore possibile. Anche se questa è la tecnica più veloce non trova la soluzione ottimale nell’insieme delle soluzioni.Invece l'algoritmo di backtracking è decisamente più lento rispetto a quello greedy, tuttavia seleziona la soluzione migliore dell'insieme delle soluzioni.Se in un problema di ottimizzazione, si pone come obiettivo il tempo di ricerca, la migliore è la tecnica greedy. Se invece si vuole ottenere tra tutte le soluzioni, quella più vantaggiosa allora bisogna prediligere il backtracking a discapito del tempo di ricerca della soluzione.Fornire la specifica sintattica e semantica degli operatori costrBinAlbero e cancSottoBinAlbero per la struttura dati Alberi binariSINTATTICACOSTRBINALBERO : (ALBEROBIN,ALBEROBIN) 🡪 ALBEROBINCANCSOTTOBINALBERO : (NODO,ALBEROBIN) 🡪 ALBEROBIN

SEMANTICA

Immagine che contiene testo

Descrizione generata automaticamenteSpiegare la realizzazione di alberi binari mediante vettore e collegata con cursori, fornendo vantaggi e svantaggi di ognunaUna possibile rappresentazione di un albero binario è quella sequenziale mediante vettore. la radice è in prima posizione; per il generico nodo p memorizzato in posizione i, se esiste il figlio sinistro è memorizzato in posizione 2\*i, se esiste il figlio destro è memorizzato in posizione 2\*i+1Immagine che contiene testo

Descrizione generata automaticamente

Se un albero è incompleto rappresenta uno spreco di memoria e per avvalorare quel nodo vuoto dell'array bisogna aggiungere il campo booleano true se presente altrimenti false.

Tuttavia è immediato verificare che:° alberi binari non completi vengono rappresentati con spreco di memoria è imposto un limite massimo per il numero di nodi dell'albero° le operazioni di aggiunta ed eliminazione di nodi o di sottoalberi comportano diversi spostamenti nell'arrayLA RAPPRESENTAZIONE COLLEGATA DI UN ALBERO

Immagine che contiene testo

Descrizione generata automaticamente

Un primo metodo richiede di utilizzare un array, in modo che ad ogni nodo dell'albero corrisponda una componente dell'array in cui sono memorizzate le informazioni (nodo, riferimento al figlio sinistro, riferimento al figlio destro).il riferimento è il valore dell'indice in corrispondenza del quale si trova la componente che corrisponde al figlio sinistro o destro. Se il figlio non esiste il riferimento ha valore 0.Fornire in pseudocodice l’algoritmo di visita in post-ordine per alberi binariVISITA POSTORDINE L'ALBERO BINARIO Tse l'albero non è vuoto, allora1-visita in postordine il sottoalbero sinistro di T2-visita in postordine il sottoalbero destro di T3-analizza la radicefineesempio:Sia un albero binario che ha dei caratteri nei nodiImmagine che contiene testo

Descrizione generata automaticamenteDescrivere come un albero binario può essere utilizzato per rappresentare una coda con priorità, descrivendo inoltre come si effettuano le operazioni di inserimento e cancellazione di elementiRAPPRESENTAZIONE CON ALBERI BINARIgli elementi di una coda con priorità C possono essere memorizzati nei nodi di un albero binario B che deve avere le proprietà:° l'albero B è quasi perfettamente bilanciato° se k è il livello massimo delle foglie, allora B ha esattamente 2k-1 nodi di livello minore di k;° tutte le sue foglie di livello k sono addossate a sinistra;° l'albero B è parzialmente ordinato° ogni nodo contiene un elemento di C che è maggiore di quello del padre per realizzare gli operatori si osservi che:- MIN restituisce il contenuto della radice dell'albero B;-INSERISCI deve inserire una nuova foglia in modo da mantenere verificata la proprietà (1) e quindi far “salire” l'elemento introdotto fino a verificare la proprietà (2);-CANCELLAMIN prevede la cancellazione della foglia di livello massimo più a destra, in modo da mantenere verificata la proprietà (1) ed il reinserimento del contenuto della foglia cancellata nell'albero partendo dalla radice e facendolo "scendere" in modo che l'albero cosi modificato verifichi anche la proprietà (2).

Fornire la specifica di problema di ottimizzazioneAlle soluzioni ammissibili è associata una misura (costo, obiettivo): risolvere il problema non significa trovare una qualunque soluzione, ma la migliore soluzione secondo la misura o criterio di preferenza fissato.Immagine che contiene testo

Descrizione generata automaticamente

Facendo riferimento ad uno Specifico problema di ricerca spiegare ed illustrare l’esecuzione di una strategia di backtracking applicata ad una istanza di quel problema.ESEMPIO:PROBLEMA DELLE 8 REGINE CON ALGORITMO DI BACKTRACKING

come si è detto possiamo esprimere la soluzione attraverso una permutazione dei numeri da 1 a 8. la soluzione si può esprimere mediante il vettore v definito come segue:TYPE Posizione=[1..8];

VETTORE=Array [1..8] of posizione;VAR V:VETTORE;Immagine che contiene testo

Descrizione generata automaticamenteI VINCOLI IMPOSTI SONO:1) Le componenti di v devono costituire una permutazione da 1 a 8; per ogni coppia di indici i e j vale V[i] ≠ V[j]2) Non è possibile avere due elementi sulla stessa diagonale, cioè:

(i-j) ≠ (k-1)(i+j) ≠(k+1)Il sottoprogramma verifica ha il compito di esaminare il vettore v e se il vettore viola uno dei vincoli rende true la variabile 8,altrimenti se i vincoli sono ambedue soddisfatti a = false.Spiegare cosa è un problema di ottimizzazione e quali tecniche si possono adottare per risolverloProblemi di ottimizzazione nei quali alle soluzioni ammissibili è associata una misura (o costo, o obiettivo) e si vuole trovare una soluzione ottima. Cioè una soluzione ammissibile la cui misura sia minima o massima (ad esempio:trovare una foglia di un albero radicato il cui livello sia massimo).Tecnica di progetto🡺 Successivamente, si cerca di applicare cene tecniche di progetto di algoritmi per rendere gli algoritmi più veloci. Le cinque tecniche principali sono: divide et impera, backtrack, greedy, programmazione dinamica e ricerca locale. La prima di queste tecniche è già stata implicitamente introdotta, e consiste nel partizionare il problema in sotto problemi più piccoli, risolvibili indipendentemente, le cui soluzioni possono essere ricombinate per ottenere la soluzione del problema di partenza (ad esempio. la procedura di ricerca binaria)Riportare l'albero binario di ricerca corrispondente ad una coda con priorità dove aver inserito nell'ordineI seguenti elementi: 17, 26, 8, 50, 16, 19, 93, 4, 9 e 55. Poi illustrare il processo di inserimento del 7.Dopo l'inserimento del 7 illustrare il processo di rimozione del minimo.Immagine che contiene testo

Descrizione generata automaticamente

Il 7 viene inserito come figlio dx di 17. Rispettando la coda con priorità e quindi la sua relazione d'ordine si fa salire il 7 fino a quando...Immagine che contiene testo

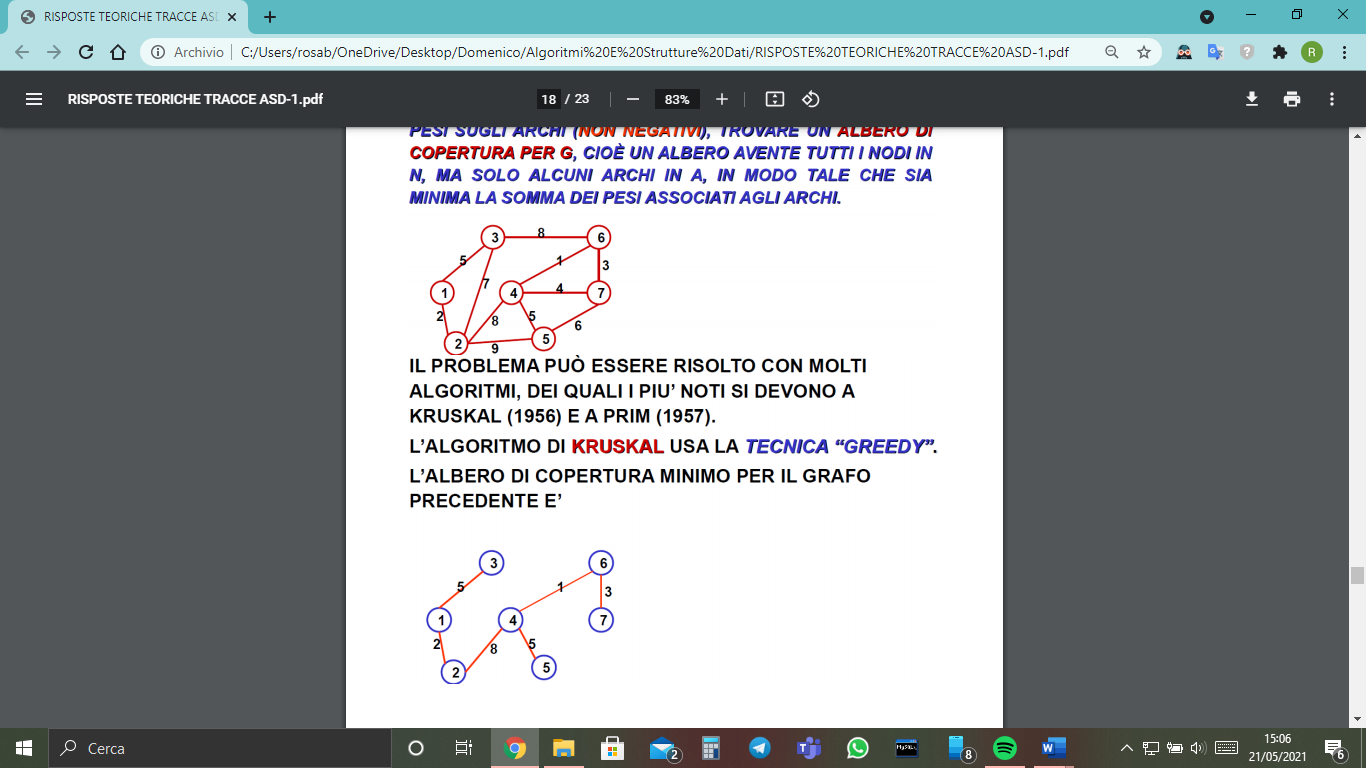
Descrizione generata automaticamente

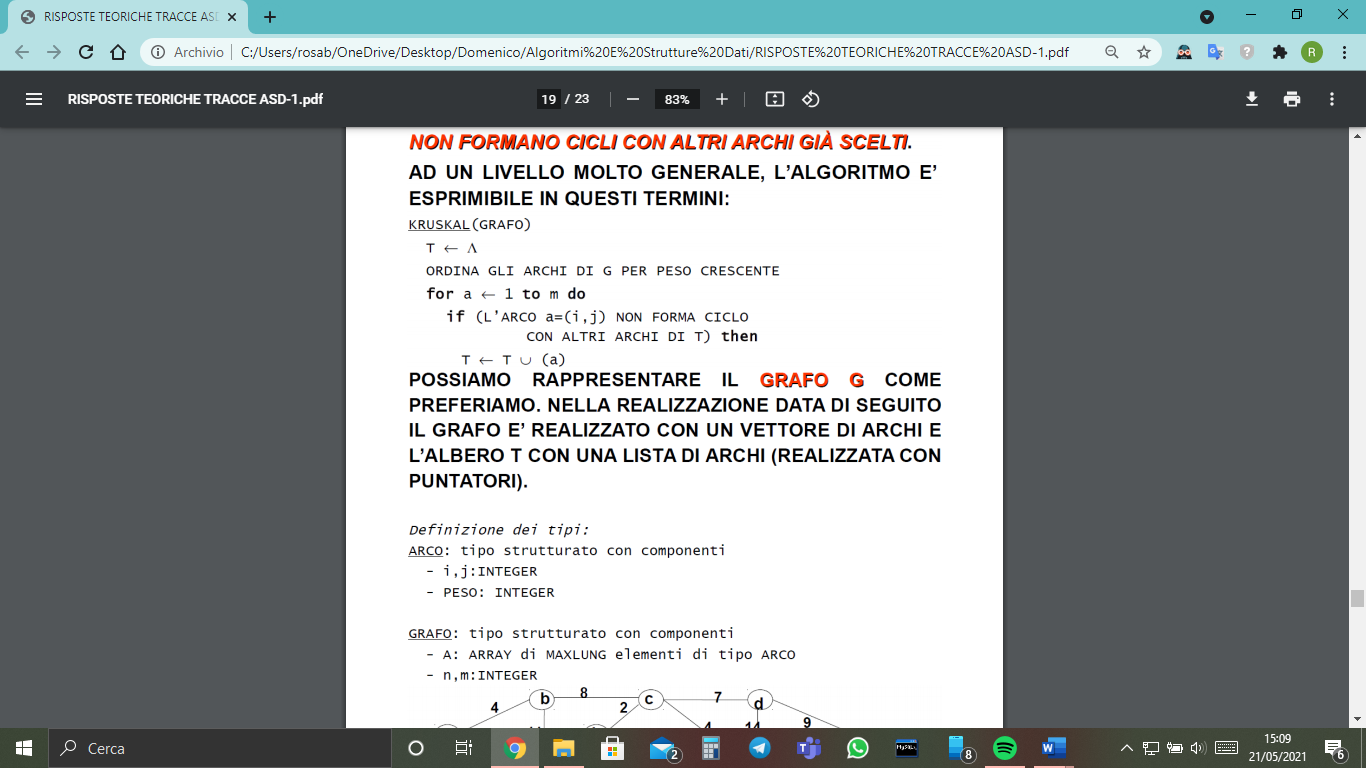
Facendo riferimento ad uno specifico problema di ottimizzazione spiegare ed illustrare l'esecuzione di una strategia greedy applicata ad una istanza di quel problema.UN ALTRO PROBLEMA: PROBLEMA DEL MINIMO ALBERO DI COPERTURADato un grafo non orientato e connesso G=(N,A), con pesi sugli archi (non negativi), trovare un albero di copertura per G, cioè un albero avente tutti i nodi in N, ma solo alcuni archi in A, in modo tale che sia minima la somma dei pesi associati agli archi.

Immagine che contiene testo

Descrizione generata automaticamente

Il problema può essere risolto con molti algoritmi, dei quali i più noti si devono a Kruskal (1956) e a Prim (1951).l'algoritmo di Kruskal usa la tecnica “greedy”.l'albero di copertura minimo per il grafo precedente è:



L'algoritmo di Kruskal, dopo aver ordinato gli archi secondo i pesi crescenti, li esamina in tale ordine, inserendoli nella soluzione se non formano cicli con altri archi già scelti. Ad un livello molto generale, l'algoritmo è esprimibile in questi termini:

Possiamo rappresentare il grafo G come preferiamo. nella realizzazione data di seguito il grafo è realizzato con un vettore di archi e l'albero T con una lista di archi (realizzata con puntatori).Immagine che contiene testo

Descrizione generata automaticamente

ECCO L'ORDINE IN CUI L'ALGORITMO CONSIDERA GLI ARCHI.Una volta che si considera l'arco se questo unisce due alberi distinti nella foresta, l’ arco viene aggiunto alla foresta e i due alberi fusi in unoImmagine che contiene testo

Descrizione generata automaticamente

Nell'algoritmo di Kruskal la costruzione di T avviene per unione di componenti connesse rappresentabili come insiemi disgiunti (due componenti connesse si fondono in una con l’aggiunta di un nuovo arco)Come è noto, la struttura mfset (merge-find set) è una partizione di un insieme in sottoinsiemi disgiunti detti componenti. Dunque è possibile utilizzare una struttura dati di tipo mfset per costruire T.

Immagine che contiene testo

Descrizione generata automaticamente

Spiegare la strategia di risoluzione per il problema della ricerca del cammino minimo in un grafo; Adottata da un algoritmo a scelta del candidatoSi basa sull'idea di calcolare, in ordine crescente, la lunghezza dei cammini minimi da r a tutti i nodi del grafo.Indichiamo con S l'insieme dei nodi di cui, ad un dato istante è già calcolato la lunghezza del cammino minimo da r.Utilizziamo un vettore DIST con tante componenti quanti sono i nodi del grafo, in modo che DIST(i) rappresenti la lunghezza del cammino minimo tra quelli che vanno da r ad i passando solo per nodi contenuti in S (a parte i stesso). L'ipotesi di fondo è che le distanze siano interi positivi. Osserviamo che se il prossimo cammino minimo da generare C è da r al nodo u, tutti i nodi sono in S.Infatti se un nodo k di C non appartenesse a S vi sarebbe un cammino da r a un nodo k non contenuto in S di lunghezza minore a quella di C, contraddicendo l'ipotesi che il prossimo cammino da generare sia C.la lunghezza di C e il nodo u sono facilmente individuabili; basta calcolare il valore minimo di DIST(i) per i∉S

Individuato u si inserisce in S e si aggiorna DIST per i nodi che ∉S. In particolare, se per un certo nodo z connesso a u da <u,z> con etichetta E, la somma DIST(u)+E è minore di DIST(z) allora a DIST(z) va assegnato il nuovo valore DIST(u)+E

Viene generato un albero di copertura T, radicato in r, che include un cammino da r ad ogni altro nodo .

L’albero radicato in T può essere rappresentato con un vettore di padri, inizializzato ad un albero “fittizio” in cui tutti i nodi sono figli di r connessi ad un arco fittizio etichettato con un valo maggiore di tutte le altre etichette (Maxint)Una soluzione ammissibile T è ottima se e solo se:

DIST(i)+Cij= DIST(j) ∀(i,j)∈T & DIST(i)+Cij ≥ DIST(j) ∀arco(i,j)∈ A

Immagine che contiene testo

Descrizione generata automaticamente

DIVIDE ET IMPERA ALGO NUM CONSECUTIVIData una sequenza di n numeri interi (x1,…….,xn) diciamo che (xi,xi+1) è una coppia di numeri consecutivi se xi+1=xi+1. Ad esempio nella sequenza (12,13,24,25,26,35,67) ci sono 3 coppie di numeri consecutivi: (12,13), (24,25), (25,26). Scrivere in pseudocodice un algoritmo che utilizzi la tecnica *divide-et-impera* e che calcoli quante coppie di numeri consecutivi sono contenute in una sequenza di n numeri interi (x1,……,xn) ricevuta in input

-NUM SUCC. ALGORITMO DIVIDE ET IMPERA (INPUT: ARRAY[])BEGININT DIM = DIM ARRAYi=0MENTRE i<DlMSE ARRAY[i] == SUCCESSIVO IN ARRAY -1INSERISCI ARRAY[i] E SUCCESSIVO IN ARRAY IN INSIEMEINCREMENTA iFINERETURN INSIEME

- SUCCESSIVO IN ARRAY -1 (INPUT ELEM, ARRAY)INT DIM = DIM ARRAYINT i=0MENTRE i<DIMSE ELEMENTO IN POS i È = ELEM+1RESTITUISCI ELMEN IN POS i+1ALTRIMENTI SE i==DIM-1RESTITUISCI NIL(NIL=-1)

Elencare il problema dello zaino

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Elencare le tre complessità esistenti

Caso ottimo: Si ottiene considerando, a parità di dimensione dei dati, la configurazione che dà luogo al minimo tempo di esecuzione. Tale complessità è per lo più di interesse secondario anche se è abbastanza facile da determinare.

Caso medio: si riferisce al tempo di calcolo mediato su tutte le possibili configurazioni dei dati, sempre per una data dimensione n

Caso pessimo: Si intende la complessità relativa a quella particolare occorrenza iniziale dei dati per cui l’algoritmo ha comportamento pessimo. Tale funzione di complessità fornisce un limite superiore alla complessità, entro cui il funzionamento dell’algoritmo è sempre garantito.