

Grafi (I parte)

Progettazione di Algoritmi a.a. 2021-22

Matricole congrue a 1

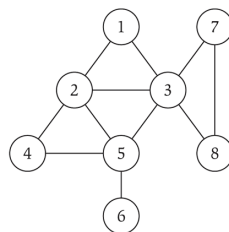
Docente: Annalisa De Bonis

1

1

Grafi non direzionati

- **Grafi non direzionati.** $G = (V, E)$
 - V = insieme nodi.
 - E = insieme archi.
 - Esprime le relazioni tra coppie di oggetti.
 - Parametri del grafo: $n = |V|$, $m = |E|$.



$$V = \{ 1, 2, 3, 4, 5, 6, 7, 8 \}$$

$$E = \{ 1-2, 1-3, 2-3, 2-4, 2-5, 3-5, 3-7, 3-8, 4-5, 5-6, 7-8 \}$$

$$n = 8$$

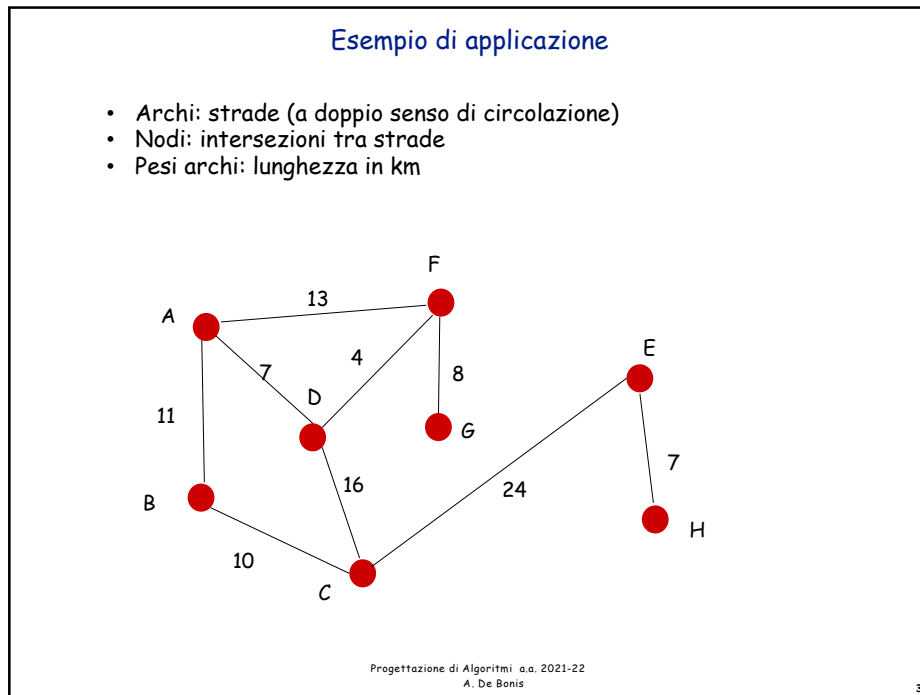
$$m = 11$$

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

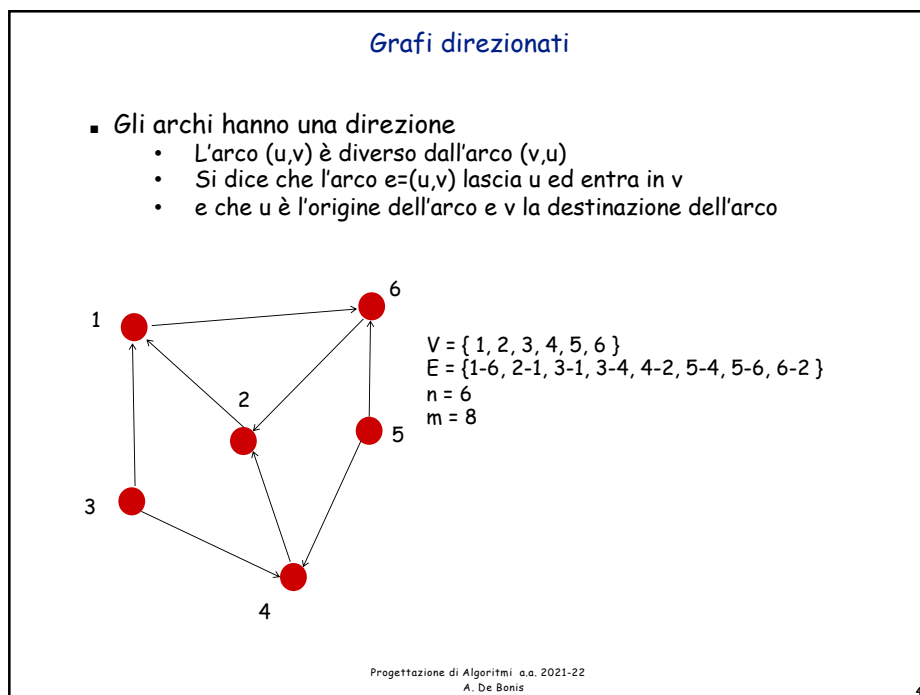
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

2

2



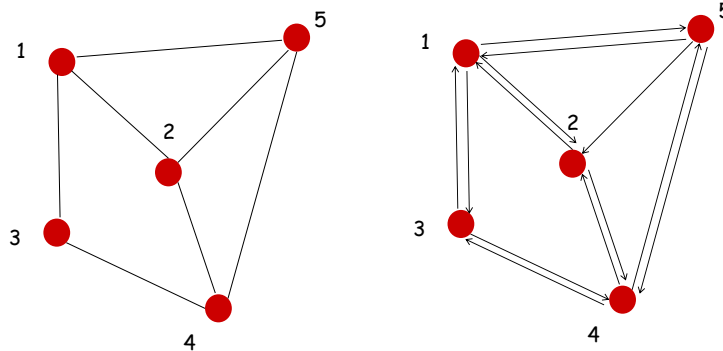
3



4

Grafi direzionati

- **Grafi non direzionati** $G = (V, E)$ possono essere visti come un caso particolare degli archi direzionati in cui per ogni arco (u,v) c'è l'arco di direzione opposta (v,u)



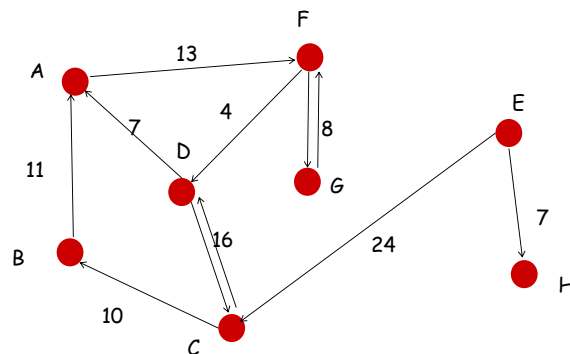
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

5

5

Esempio di applicazione

- Archi: strade (a senso unico di circolazione)
- Nodi: intersezioni tra strade
- Pesi archi: lunghezza in km



6

6

Alcune applicazioni dei grafi

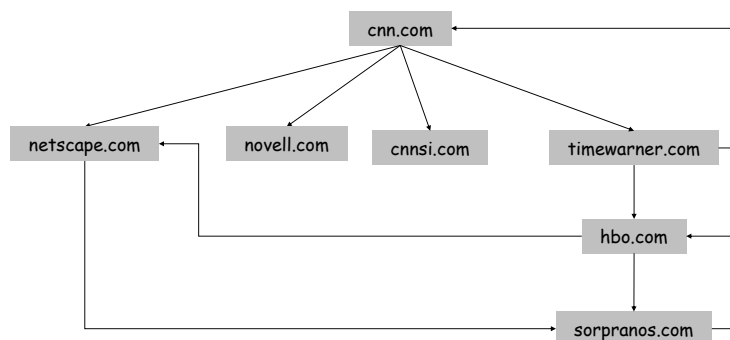
- Rete di amicizia su un social network: ogni utente è un nodo; ogni volta che due utenti diventano amici, si crea un arco del grafo.
- Google maps: i nodi rappresentano città, intersezioni di strade, siti di interesse, ecc. e gli archi rappresentano le connessioni dirette tra i nodi.
 - La rappresentazione mediante un grafo permette di trovare il percorso più corto per andare da un posto all'altro mediante un algoritmo.
- World Wide Web: le pagine web sono i nodi e il link tra due pagine è un arco. Google utilizza questa rappresentazione per esplorare il World Wide Web

7

7

World Wide Web

- Web graph.
 - Nodo: pagina web.
 - Edge: hyperlink da una pagina all'altra.



Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

8

8

Ecological Food Web

- Food web graph.
 - Nodo = specie
 - Arco dalla preda al predatore.

Reference: <http://www.twingroves.district96.k12.il.us/Wetlands/Salamander/SalGraphics/salfoodweb.gif>

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

9

Alcune applicazioni dei grafi

<i>Graph</i>	<i>Nodi</i>	<i>Archi</i>
trasporto	intersezioni di strade	strade
trasporto	aeroporti	voli diretti
comunicazione	computer	cavi di fibra ottica
World Wide Web	web page	hyperlink
rete sociale	persone	relazioni
rete del cibo	specie	predatore-preda
scheduling	task	vincoli di precedenza
circuiti	gate	wire

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

10

Terminologia

- Consideriamo due nodi u e v di un grafo G connessi dall'arco $e = (u,v)$
- Si dice che
 - u e v sono adiacenti
 - u e v sono le estremità dell'arco (u,v)
 - l'arco (u,v) incide sui vertici u e v
 - u è un nodo vicino di v
 - v è un nodo vicino di u
- Dato un vertice u di un grafo G
 - grado di u = numero archi incidenti su u
 - è indicato con $\text{deg}(u)$

11

11

Numero di archi di un grafo non direzionato

m = numero di archi di G ;

n = numero di nodi di G .

1. La somma di tutti i gradi dei nodi di G è $2m$: $\sum_{u \in V} \text{deg}(u) = 2m$

Degree (grado) = numero di vicini di u

Dim. Ciascun arco incide su due vertici e quindi viene contato due volte nella sommatoria in alto. L'arco (x,y) è contato sia in $\text{deg}(x)$ che in $\text{deg}(y)$.

2. Il numero m di archi di un grafo G non direzionato è al più $n(n-1)/2$.

Dim. Il numero di coppie **non ordinate** distinte che si possono formare con n nodi è $n(n-1)/2$. Posso scegliere il primo nodo dell'arco in n modi e il secondo in modo che sia diverso dal primo nodo, cioè in $n-1$ modi. Dimezzo in quanto l'arco (u,v) è uguale all'arco (v,u)

12

12

Numero di archi di un grafo direzionato

m = numero di archi di G ;

n = numero di nodi di G

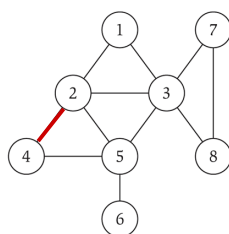
Il numero m di archi di G è al più n^2

Dim. Il numero di coppie **ordinate** distinte che si possono formare con n nodi è n^2 . Posso scegliere il primo nodo dell'arco in n modi e il secondo in altri n modi (se ammettiamo archi con entrambe le estremità uguali).

13

Graph Representation: Adiacenza Matrix

- **Matrice di adiacenza.** Matrice $n \times n$ con $A_{uv} = 1$ se (u, v) è un arco.
 - Spazio proporzionale a n^2 .
 - Controllare se (u, v) è un arco richiede tempo $\Theta(1)$.
 - Identificare tutti gli archi richiede tempo $\Theta(n^2)$.

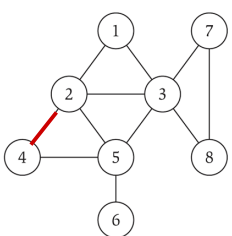


	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

14

Rappresentazione di un grafo: liste di adiacenza

- **Liste di adiacenza.** Array di liste in cui ogni lista è associata ad un nodo.
 - Ad ogni arco corrisponde un elemento della lista.
 - Se esiste l'arco (u,v) allora la lista associata ad u contiene v
 - In un grafo non direzionato l'arco (u,v) corrisponde ad un elemento della lista associata ad u e ad un elemento della lista associata a $v \rightarrow$ somma lunghezze liste = $2m$
 - In un grafo direzionato l'arco (u,v) corrisponde ad un elemento della lista associata ad $u \rightarrow$ somma lunghezze liste = m
 - Spazio proporzionale a $m + n$.
 - Controllare se (u, v) è un arco richiede tempo $O(\deg(u))$.
 - Individuare tutti gli archi richiede tempo $\Theta(m + n)$.



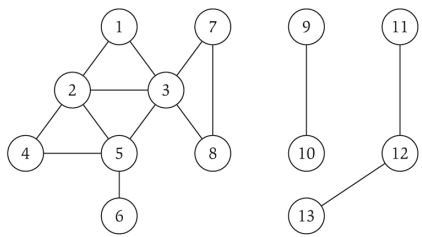
1	2	3					
2	1	3	4	5			
3	1	2	5	7	8		
4	2	5					
5	2	3	4	6			
6	5						
7	3	8					
8	3	7					

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

15

Percorsi e connettività

- **Def.** Un percorso in un grafo non direzionato $G = (V, E)$ è una sequenza P di nodi $v_1, v_2, \dots, v_{k-1}, v_k$ con la proprietà che ciascuna coppia di vertici consecutivi v_i, v_{i+1} è unita da un arco in E .
- **Def.** Un percorso è **semplice** se tutti i nodi sono distinti.
- **Def.** Un grafo non direzionato è **connesso** se per ogni coppia di nodi u e v , esiste un percorso tra u e v .



Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

16

Applicazione del concetto di percorso

- **Esempi:**
Web graph. Voglio capire se è possibile, partendo da una pagina web e seguendo gli hyperlink nelle pagine via via attraversate, arrivare ad una determinata pagina

```
graph TD; cnn[cnn.com] --> netscape[netscape.com]; cnn --> novell[novell.com]; cnn --> cnnsi[cnnsi.com]; cnn --> timewarner[timewarner.com]; timewarner --> cnn; timewarner --> hbo[hbo.com]; hbo --> timewarner; hbo --> sorpranos[sorpranos.com]; netscape --> sorpranos; novell --> sorpranos;
```

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

17

17

Applicazione del concetto di percorso

In alcuni casi può essere interessante scoprire il percorso più corto, cioè composto dal minimo numero di archi, tra due nodi.

Esempio:

- Grafo : rete di trasporti dove i nodi sono gli aeroporti e gli archi i collegamenti diretti tra aeroporti.
- Voglio arrivare da Napoli a New York facendo il minimo numero di scali.

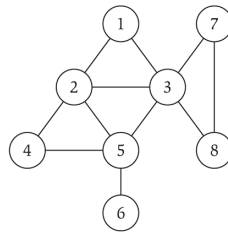
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

18

18

Cicli

- **Def.** Un ciclo è un percorso $v_1, v_2, \dots, v_{k-1}, v_k$ in cui $v_1 = v_k$, $k > 2$.
- **Def.** Un ciclo $v_1, v_2, \dots, v_{k-1}, v_1$ è **semplice** se i primi $k-1$ nodi del ciclo sono tutti distinti tra di loro



ciclo (semplice) $C = 1-2-4-5-3-1$

ciclo (non semplice) $C' = 1-3-7-8-3-5-2-1$

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

19

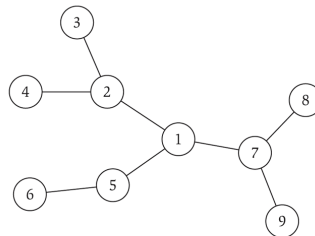
19

Alberi

- **Def.** Un grafo non direzionato è un **albero (tree)** se è connesso e non contiene cicli
- **Teorema.** Sia G un grafo non direzionato con n nodi. Ogni due delle seguenti affermazioni implica la restante affermazione.

- $1 \text{ e } 2 \implies 3$; $1 \text{ e } 3 \implies 2$; $2 \text{ e } 3 \implies 1$

1. G è connesso.
2. G non contiene cicli.
3. G ha $n-1$ archi.



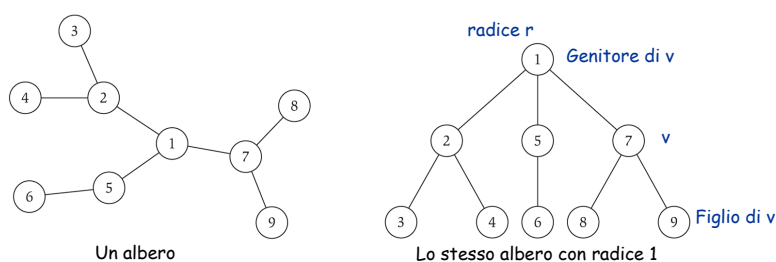
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

20

20

Alberi con radice

- **Albero con radice.** Dato un albero T , si sceglie un nodo radice r e si considerano gli archi di T come orientati a partire da r
- Dato un nodo v di T si dice
 - **Genitore di v :** il nodo che w precede v lungo il percorso da r a v (v viene detto figlio di w)
 - **Antenato di v :** un qualsiasi nodo w lungo il percorso che va da r a v (v viene detto discendente di w)
- **Foglia:** nodo senza discendenti

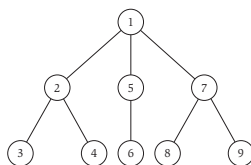


Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

21

Alberi con radice

- Scegliere un nodo come radice, rende più semplice dimostrare la seguente affermazione
- **se G è connesso e non contiene cicli**, in altre parole, **se G è un albero** allora il numero di archi è $n-1$ (dove n è il numero di nodi).
- **Dim.**
- Per ogni nodo diverso dalla radice c'è un arco distinto che lo connette al proprio padre \rightarrow numero di archi $\geq n-1$
- Per ogni arco c'è un nodo distinto (non radice) che è congiunto al padre da quell'arco \rightarrow numero di archi $\leq n-1$
- le due disequaglianze \rightarrow numero di archi = $n-1$



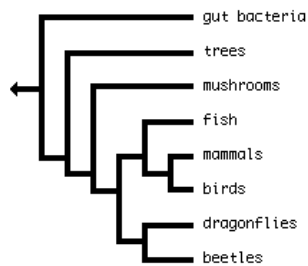
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

22

22

Importanza degli alberi: rappresentano strutture gerarchiche

- **Alberi filogenetici.** Descrivono la storia evolutiva delle specie animali.



La filogenesi afferma l'esistenza di una specie ancestrale che diede origine a mammiferi e uccelli ma non alle altre specie rappresentate nell'albero (cioè, mammiferi e uccelli condividono un antenato che non è comune ad altre specie nell'albero). La filogenesi afferma inoltre che tutti gli animali discendono da un antenato non condiviso con i funghi, gli alberi e i batteri, e così via.

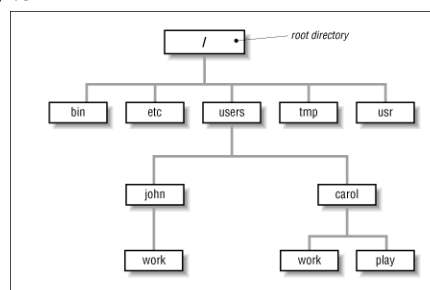
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

23

23

Importanza degli alberi: rappresentano strutture gerarchiche

- **File system.** Un file system tipicamente consiste di file organizzati in gruppi chiamati directory.
 - Una directory può contenere file e altre directory,
 - Un file system gerarchico è organizzato secondo una struttura gerarchica ad albero con radice
 - nodi interni: directory
 - foglie: file



Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

24

24

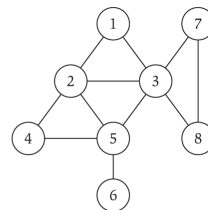
Visite di grafi

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

25

Connettività

- **Problema della connettività tra s e t .** Dati due nodi s e t , esiste un percorso tra s e t ?
- **Problema del percorso più corto tra s e t .** Dati due nodi s e t , qual è la lunghezza del percorso più corto tra s e t ?
- **Applicazioni.**
 - Attraversamento di un labirinto.
 - Erdős number.
 - Minimo numero di dispositivi che devono essere attraversati dai dati in una rete di comunicazione per andare dalla sorgente alla destinazione
 - Minimo numero di scali in un viaggio aereo



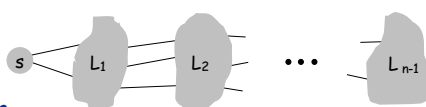
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

26

26

Breadth First Search (visita in ampiezza)

- **BFS.** Esplora il grafo a partire da una sorgente s muovendosi in tutte le possibili direzioni e visitando i nodi livello per livello (N.B.: il libro li chiama layer e cioè strati).
- I layer sono descritti di seguito



- **BFS algorithm.**
 - $L_0 = \{s\}$.
 - $L_1 =$ tutti i vicini di s .
 - $L_2 =$ tutti i nodi che non appartengono a L_0 o L_1 , e che sono uniti da un arco ad un nodo in L_1 .
 - $L_{i+1} =$ tutti i nodi che non appartengono agli strati L_0, L_1, \dots, L_i e che sono uniti da un arco ad un nodo in L_i .

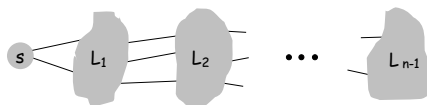
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

27

27

Breadth First Search

- **distanza tra u e v** = lunghezza del percorso più corto tra u e v
- **Teorema.** Per ogni i , L_i consiste di tutti i nodi a distanza i da s . Di conseguenza, c'è un percorso da s a t se e solo se t appare in qualche livello.



L_1 : livello dei nodi a distanza 1 da s

L_2 : livello dei nodi a distanza 2 da s

...

L_{n-1} : livello dei nodi a distanza $n-1$ da s

Il teorema si può dimostrare in modo molto semplice usando l'induzione

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

28

28

Breadth First Search

- **Teorema.** Per ogni i , L_i consiste di tutti i nodi a distanza i da s . Di conseguenza, c'è un percorso da s a t se e solo se t appare in L_i , per un certo $i \in \{0, 1, \dots, j\}$.

Dim. per induzione sull'indice del layer:

Base induttiva: per $j=0$, la tesi è vera perché $L_0 = \{s\}$ e s è l'unico nodo a distanza 0 da se stesso

Passo induttivo: Supponiamo vera la tesi fino ad un certo j e dimostriamo che è vera per $j+1$. Assumiamo quindi che per $i=0, 1, \dots, j$, il layer L_i consista di tutti i nodi a distanza i da s .

Per def. di L_{j+1} , un nodo u è in L_{j+1} se e solo se valgono le seguenti 1 e 2:

1. u non appartiene a L_0, L_1, \dots, L_j
2. u è unito da un arco ad un nodo di L_j

un nodo u soddisfa la 1 e la 2 \leftrightarrow la distanza di u da s è $j+1$
(ricordiamo che stiamo assumendo l'ipotesi induttiva)

29

Breadth First Search

dim. che un nodo u soddisfa sia la 1 che la 2 \leftrightarrow c

- è immediato vedere che vale \rightarrow
- vediamo perché:
- per la 2 esiste un arco da un certo nodo z di L_j ad u e siccome per ipotesi induttiva, z è a distanza j da s allora **esiste un percorso di lunghezza $j+1$ da s ad u**
- per la 1 il vertice u non è in $L_0 \cup L_1 \cup \dots \cup L_j$ e di conseguenza **u è a distanza $> j$ da s** (infatti per ipotesi induttiva tutti i nodi a distanza $\leq j$ da s sono in $L_0 \cup L_1 \cup \dots \cup L_j$)
- le affermazioni in rosso, implicate rispettivamente dalla 1 e dalla 2, implicano che esiste un percorso da s ad u di $j+1$ archi e questo è il più corto possibile \rightarrow la distanza da u ad s è $j+1$

30

Breadth First Search

- dimostriamo che vale \leftarrow
- Supponiamo che u sia a distanza $j+1$ da s e dimostriamo che valgono la 1 e la 2.
- Per ipotesi induttiva la 1 deve essere necessariamente soddisfatta (infatti per ipotesi induttiva L_0, L_1, \dots, L_j contengono solo nodi a distanza $\leq j$ da s)
- Dimostriamo che vale anche la 2. Sia P un percorso di $j+1$ archi da s ad u . Questo percorso esiste dal momento che la distanza da s di u è proprio $j+1$. Indichiamo con v il predecessore di u lungo P (P termina con l'arco (v,u)).
- Il sottopercorso P' di P che arriva fino a v ha lunghezza j e di conseguenza la distanza di v da s è $\leq j$.
- Facciamo vedere che la distanza di v da s è esattamente j .
Se per assurdo questa distanza fosse $< j$ allora esisterebbe un percorso P'' da s a v di lunghezza $< j$ e il percorso da s ad u ottenuto concatenando a P'' l'arco (v,u) avrebbe lunghezza $< j+1$ contraddicendo l'ipotesi che u fosse a distanza $j+1$ da s .
- Siccome la distanza di v da s è esattamente j allora per ipotesi induttiva $v \in L_j$ e di conseguenza l'arco (v,u) unisce u ad un nodo in $L_j \rightarrow u$ soddisfa la 2.

Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

31

31

Breadth First Search

Pseudocodice (schema dell'algoritmo)

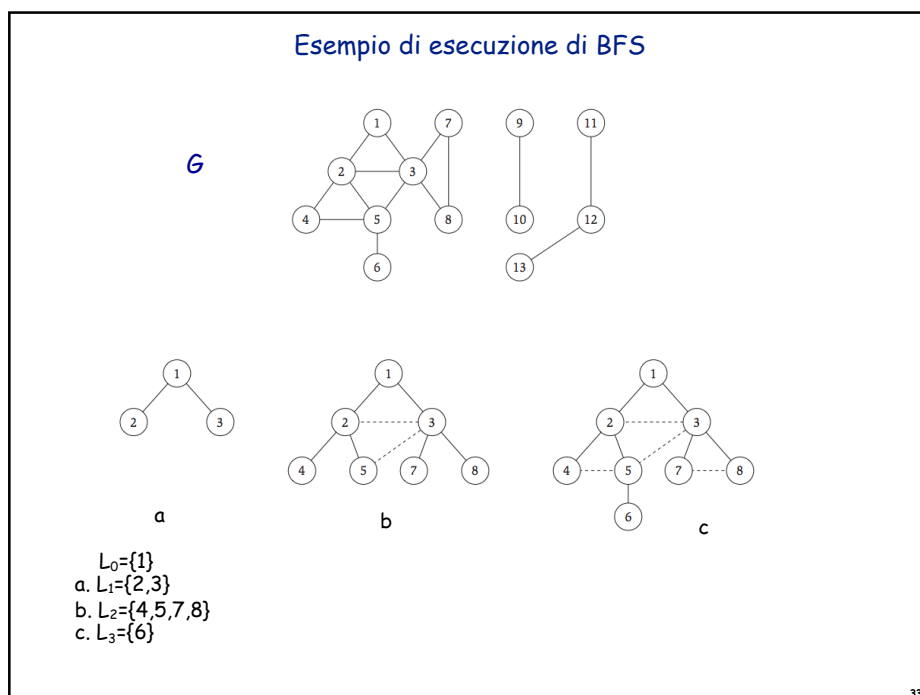
1. **BFS**(s)
2. $L_0 = \{ s \}$
3. **For**($i=0; i \leq n-2; i++$)
4. $L_{i+1} = \emptyset$;
5. **Foreach** nodo u in L_i
6. **Foreach** nodo v adiacente ad u
7. **if**(v non appartiene ad L_0, \dots, L_{i+1})
8. $L_{i+1} = L_{i+1} \cup \{ v \}$
9. **EndIf**
10. **Endforeach**
11. **Endforeach**
12. **Endfor**

- Occorre un modo per capire se un nodo è già stato visitato in precedenza. Il tempo di esecuzione dipende dal modo scelto, da come è implementato il grafo e da come sono rappresentati gli insiemi L_i che rappresentano i livelli

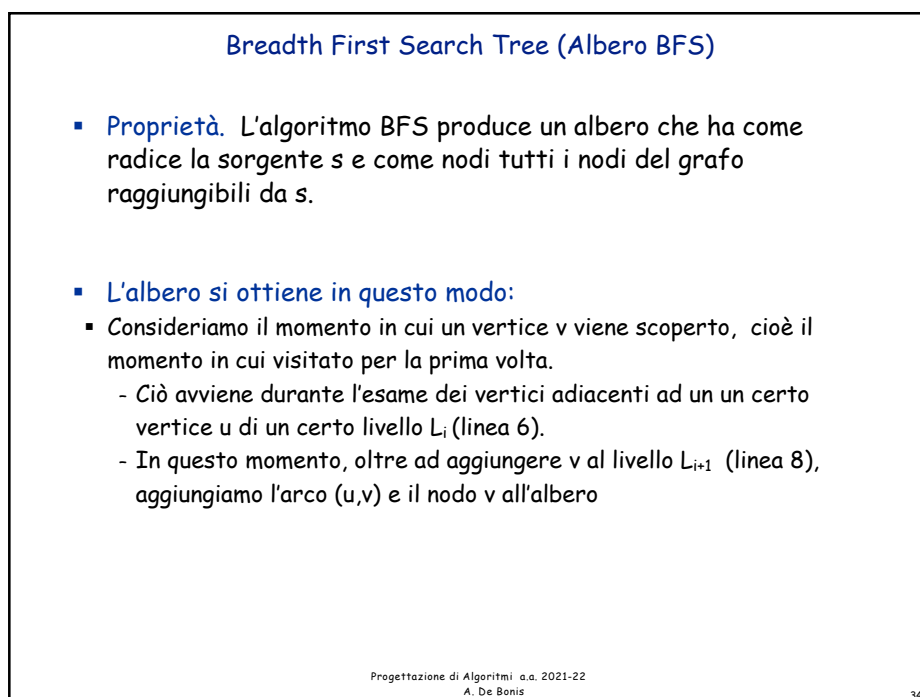
Progettazione di Algoritmi a.a. 2021-22
A. De Bonis

32

32



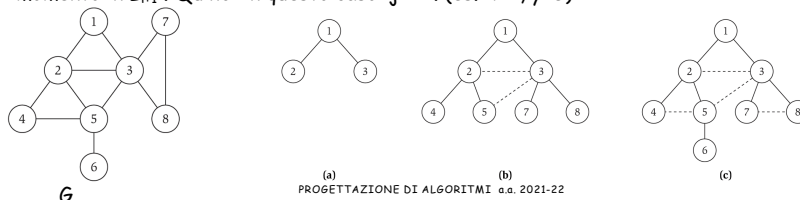
33



34

Breadth First Search Tree

- **Proprietà.** Si consideri un'esecuzione di BFS su $G = (V, E)$, e sia (x, y) un arco di G . I livelli di x e y differiscono di al più di 1.
- **Dim.** Sia L_i il livello di x ed L_j quello di y . Supponiamo senza perdere di generalità che x venga scoperto prima di y cioè che $i \leq j$. Consideriamo il momento in cui l'algoritmo esamina gli archi incidenti su x .
 - **Caso 1.** Il nodo y è stato già scoperto:
Siccome per ipotesi y viene scoperto dopo x allora sicuramente y viene inserito
 - a) o nel livello i dopo x , se y è adiacente a qualche nodo nel livello $i-1$ (es. $x=2, y=3$).
 - b) o nel livello $i+1$, se è adiacente a qualche nodo del livello i esaminato nel **For each** alla linea 5 prima di x . (es. $x=3, y=5$)
 Quindi in questo caso si ha $j=i$ oppure $j=i+1$.
 - **Caso 2.** Il nodo y non è stato ancora scoperto:
Siccome tra gli archi incidenti su x c'è anche (x, y) allora y viene inserito in questo momento in L_{i+1} . Quindi in questo caso $j=i+1$. (es. $x=2, y=5$)



35

35

Implementazione di BFS per grafo implementato con liste di adiacenza

- Ciascun insieme L_i è rappresentato da una lista $L[i]$
 - Usiamo un array di valori booleani *Discovered* per associare a ciascun nodo il valore vero o falso a seconda che sia già stato scoperto o meno
 - Durante l'algoritmo costruiamo anche l'albero BFS
1. **BFS(s):**
 2. Poni *Discovered[s] = true* e *Discovered[v] = false* per tutti gli altri v
 3. Inizializza $L[0]$ in modo che contenga solo s
 4. Poni il contatore dei livelli $i = 0$
 5. Inizializza il BFS tree T con un albero vuoto
 6. **While** $i < n-2$ // $L[i]$ è vuota se non ci sono
 7. Inizializza $L[i+1]$ con una lista vuota // nodi raggiungibili da $L[i-1]$
 8. **Foreach** $u \in L[i]$
 9. **Foreach** arco (u, v) incidente su u
 10. **If** *Discovered[v] = false*
 11. Poni *Discovered[v] = true*
 12. Aggiungi v alla lista $L[i+1]$
 13. Aggiungi l'arco (u, v) all'albero T
 14. **Endif**
 15. **Endfor**
 16. **EndFor**
 17. $i=i+1$
 18. **Endwhile**

36

36

Implementazione di BFS per grafo implementato con liste di adiacenza

1. BFS(s):
2. Poni Discovered[s] = true e Discovered[v] = false per tutti gli altri v $O(n)$
3. Inizializza L[0] in modo che contenga solo s
4. Poni il contatore dei livelli i = 0 $O(1)$
5. Inizializza il BFS tree T con un albero vuoto $O(1)$
6. **While** i <= n-2 n volte
7. Inizializza L[i+1] con una lista vuota n-1 volte $O(n)$
8. **Foreach** u ∈ L[i] Sul totale di tutte le iterazioni del while, al più n volte
9. **Foreach** arco (u, v) incidente su u Sul totale di tutte le iterazioni del While al più 2m volte.
10. **If** Discovered[v] = false **then**
11. Poni Discovered[v] = true
12. Aggiungi v alla lista L[i+1]
13. Aggiungi l'arco (u, v) all'albero T $O(m)$
14. **Endif**
15. **Endfor**
16. **Endfor**
17. i=i+1
18. **Endwhile**

Algoritmo è $O(n+m)$

Foreach più esterno viene eseguito al più n volte in quanto ogni nodo raggiungibile da s appartiene ad una sola lista L_i
 Foreach più interno viene eseguito al più $\sum_{u \in V} \text{deg}(u) \leq 2m$ volte

37

37

Breadth First Search Tree

- **Proprietà.** Si consideri un'esecuzione di BFS su $G = (V, E)$, e sia (x, y) un arco di G . I livelli di x e y differiscono di al più di 1.
- **Dim.** Sia L_i il livello di x ed L_j quello di y . Supponiamo senza perdere di generalità che x venga scoperto prima di y cioè che $i \leq j$. Consideriamo il momento in cui l'algoritmo esamina gli archi incidenti su x .
- **Caso 1.** Il nodo y è stato già scoperto:
 Siccome per ipotesi y viene scoperto dopo x allora sicuramente y viene inserito
 a) o nel livello i dopo x , se y è adiacente a qualche nodo nel livello $i-1$ (es. $x=2, y=3$).
 b) o nel livello $i+1$, se è adiacente a qualche nodo del livello i esaminato nel **For each** alla linea 5 prima di x . Quindi in questo caso si ha $j=i$ oppure $j=i+1$. (es. $x=3, y=5$)
- **Caso 2.** Il nodo y non è stato ancora scoperto:
 Siccome tra gli archi incidenti su x c'è anche (x,y) allora y viene inserito in questo momento in L_{i+1} . Quindi in questo caso $j=i+1$. (es. $x=2, y=5$)

G

(a)

(b)

(c)

PROGETTAZIONE DI ALGORITMI a.a. 2021-22
A. DE BONIS

38

38

Implementazione di BFS con coda FIFO

L'algoritmo BFS si presta ad essere implementato con un coda
 Ogni volta che viene scoperto un nodo u , il nodo u viene inserito nella coda
 Vengono esaminati gli archi incidenti sul nodo al front della coda

BFS(s)

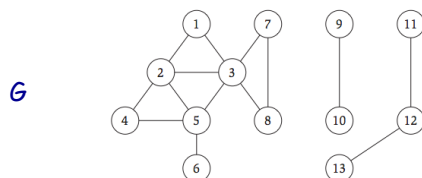
1. Inizializza Q con una coda vuota
2. Inizializza il BFS tree T con un albero vuoto
3. Poni $Discovered[s] = true$ e $Discovered[v] = false$ per tutti gli altri v
4. Inserisci s in coda a Q con una enqueue
5. While(Q non è vuota)
6. estrai il front di Q con una deque e ponilo in u
7. Foreach arco (u,v) incidente su u
8. If($Discovered[v]=false$)
9. poni $Discovered[v]=true$
10. aggiungi v in coda a Q con una enqueue
11. aggiungi (u,v) al BFS tree T
12. Endif
13. Endfor
14. Endwhile

Dimostrare per esercizio che il tempo di esecuzione è $O(n+m)$
 (svolto in classe)

39

39

Esempio di esecuzione di BFS con coda FIFO



elementi della coda durante l'esecuzione (front = elemento più a sinistra)

all'inizio $Q = 1$

dopo I iterazione del while $Q = 2\ 3$

dopo II iterazione del while $Q = 3\ 4\ 5$

dopo III iterazione del while $Q = 4\ 5\ 7\ 8$

dopo IV iterazione del while $Q = 5\ 7\ 8$

dopo V iterazione del while $Q = 7\ 8\ 6$

dopo VI iterazione del while $Q = 8\ 6$

dopo VII iterazione del while $Q = 6$

dopo VIII iterazione del while Q è vuota

- Viene estratta la sorgente e vengono inseriti i nodi del livello 1.
- Poi man mano vengono estratti i nodi del livello 1 ed inseriti quelli del livello 2.
- Quando non ci sono più elementi del livello 1, cominciano ad essere estratti i nodi del livello 2 e via via inseriti quelli del livello 3 e così via.

40

40