Tecniche Algoritmiche /3
Paradigma generativo: tecnica golosa e tecnica Divide-et-impera

Algoritmi e Strutture Dati + Lab
A.A. 15/16

Informatica
Università degli Studi di Bari “Aldo Moro”

Nicola Di Mauro

DAL PARADIGMA *GENERATIVO* SCATURISCONO TECNICHE DI PROGETTO DI ALGORITMI CHE GENERANO DIRETTAMENTE LA SOLUZIONE SENZA SELEZIONARLA TRA GLI ELEMENTI DELLO SPAZIO DI RICERCA.

IN QUESTO PARADIGMA LO SPAZIO DI RICERCA È CONSIDERATO ESCLUSIVAMENTE IN FASE DI PROGETTO DELL'ALGORITMO ALLO SCOPO DI CARATTERIZZARE LE SOLUZIONI DEL PROBLEMA E DEFINIRE UNA STRATEGIA RISOLUTIVA DIRETTA PER OGNI ISTANZA.

APPARTENGONO A QUESTO PARADIGMA LA TECNICA GOLOSA E LA DIVIDE- ET-IMPERA.

LA TECNICA GOLOSA (GREEDY)

SI APPLICA PRINCIPALMENTE A PROBLEMI DI OTTIMIZZAZIONE.

RICHIEDE CHE L'ALGORITMO ESEGUA IL PROCESSO DI COSTRUZIONE DI UN ELEMENTO DELLO SPAZIO DI RICERCA IN STADI E SI BASA SUI SEGUENTI PRINCIPI:

- **AD OGNI *STADIO i*, PER LA COMPONENTE *i-ESIMA* VIENE SCELTO IL VALORE CHE, TRA QUELLI AMMISSIBILI, RISULTA IL MIGLIORE RISPETTO DA UN DETERMINATO CRITERIO**
- **UNA VOLTA FATTA LA SCELTA PER LA i-ESIMA COMPONENTE, SI PASSA A CONSIDERARE LE ALTRE, *SENZA PIÙ TORNARE SULLA DECISIONE PRESA* .**

LO SCHEMA DI UN ALGORITMO GOLOSO (GREEDY)

PRESUPPONE CHE L'ALGORITMO ACQUISISCA LA RAPPRESENTAZIONE DI UNA ISTANZA DEL PROBLEMA E DISPONGA DI UN METODO PER ORGANIZZARE IN STADI LA COSTRUZIONE DI UN ELEMENTO DELLO SPAZIO DI RICERCA

1. **PONI i A 1 E INIZIALIZZA z**
2. **DETERMINA L'INSIEME A DEI VALORI AMMISSIBILI PER LA COMPONENTE i-ESIMA DI z E SE A ≠ ∅ SCEGLI IL MIGLIORE IN A, RISPETTO AL CRITERIO DI PREFERENZA FISSATO**
3. **SE L'i-ESIMO STADIO È L'ULTIMO ALLORA TERMINA E RESTITUISCE o(z) COME RISULTATO**
4. **ALTRIMENTI INCREMENTA i DI 1 E TORNA AL PASSO 2**

ESEMPIO:

IL PROBLEMA DELLO ZAINO (KNAPSACK)

COME È NOTO, SI DISPONE DI UN **BUDGET B** E SI DEVE CERCARE DI MASSIMIZZARE LA RENDITA SCEGLIENDO TRA **n** POSSIBILI INVESTIMENTI, CIASCUNO DEI QUALI CARATTERIZZATO DA UN PROFITTO P_i E DA UN COSTO C_i. IL PROBLEMA VIENE FORMULATO COSÌ:

DATI **2n+1** INTERI POSITIVI

$$P_1 , P_2 , \dots , P_n , C_1 , C_2 , \dots , C_n , B$$

SI VOGLIONO TROVARE n VALORI

$$X_1 , X_2 , \dots , X_n$$ TALI CHE

$X_n \in \{ 0 , 1 \}$ $(1 \leq i \leq n)$

$\sum_{i=1}^{n} P_i X_i$ SIA MASSIMA

$\sum_{i=1} C_i X_i \leq B$

CONSIDERIAMO L'ISTANZA B=5 E TRE ELEMENTI DISPONIBILI.

i	P_i	C_i	P_i / C_i
1	6	2	3
2	4	1	4
3	7	3	3.5

IL CRITERIO DI BASE PER ORDINARE LE VARIABILI È QUELLO DI VALUTARE SIA IL COSTO CHE IL PROFITTO P_i / C_i .

UNA VOLTA EFFETTUATO L'ORDINAMENTO, SI POSSONO SCEGLIERE LE VARIABILI PONENDOLE AL MASSIMO VALORE COMPATIBILE CON I VINCOLI.

IL METODO/1

- **DEFINIRE UNA FUNZIONE *ZAINOGREEDY* CHE HA COME PARAMETRI:**
  - ***IL NUMERO n DI OGGETTI DENOTATI DAI NUMERI INTERI DA 0 A n – 1***
  - ***IL VETTORE P DEI PROFITTI***
  - ***IL VETTORE C DEI COSTI***
  - ***IL BUDGET B***
- **IL RISULTATO VIENE RESTITUITO NEL VETTORE X NEL QUALE LA i-ESIMA COMPONENTE VALE 1 SE L'OGGETTO i È INCLUSO NELLA SOLUZIONE, E 0 ALTRIMENTI**

IL METODO/2

- LA FUNZIONE ORDINA IN UN VETTORE AUSILIARIO **V** GLI OGGETTI SECONDO **L'ORDINE NON CRESCENTE** DEL RAPPORTO P_i/C_i. OGNI ELEMENTO È IN CORRISPONDENZA DI UN OGGETTO E CONTIENE L'INDICE DELL'OGGETTO IN UN APPOSITO CAMPO, OLTRE AL RAPPORTO P_i/C_i AD ESSO ASSOCIATO;
- DOPO L'ORDINAMENTO, LA FUNZIONE SCANDISCE IL VETTORE **V**, ANALIZZANDO $\forall$ **i** L'OGGETTO IN **V(i)** E DECIDENDO, IN BASE ALLA COMPATIBILITÀ CON IL VINCOLO SUL BUDGET, SE INCLUDERLO O MENO NELLA SOLUZIONE.

TECNICA GREEDY: CONCETTI DI BASE

UN ALGORITMO GREEDY PERMETTE DI OTTENERE UNA SOLUZIONE MEDIANTE UNA SEQUENZA DI DECISIONI; IN OGNI PASSO VIENE PRESA LA DECISIONE CHE AL MOMENTO APPARE MIGLIORE.

QUESTA STRATEGIA EURISTICA NON GARANTISCE SEMPRE UNA SOLUZIONE OTTIMA

COME SI PUÒ DETERMINARE SE UN ALGORITMO GREEDY È IN GRADO DI TROVARE LA SOLUZIONE DI UN PROBLEMA DI OTTIMIZZAZIONE?

I PROBLEMI CHE SI PRESTANO AD ESSERE RISOLTI CON UNA STRATEGIA GREEDY PRESENTANO ALCUNE CARATTERISTICHE :

- ***LA PROPRIETÀ DELLA SCELTA GREEDY***
- ***LA SOTTOSTRUTTURA OTTIMA***

LA PROPRIETÀ DELLA SCELTA GREEDY ASSICURA CHE SI PUÒ OTTENERE UNA SOLUZIONE OTTIMA GLOBALE PRENDENDO DECISIONI CHE SONO ***OTTIMI LOCALI***.

UN PROBLEMA PRESENTA UNA ***SOTTOSTRUTTURA OTTIMA*** SE UNA SOLUZIONE OTTIMA DEL PROBLEMA CONTIENE AL SUO INTERNO UNA SOLUZIONE OTTIMA DEI SOTTOPROBLEMI.

LA TECNICA GREEDY È SPESSO UTILE PER PROBLEMI DI *"SCHEDULING"*, IN CUI SI HANNO PROGRAMMI DA ESEGUIRE SU UN "PROCESSORE" E SI VUOLE L'ORDINE DI ESECUZIONE "OTTIMO" IN BASE A UN CERTO CRITERIO.

ESEMPIO:

PROBLEMA DI SELEZIONE DI ATTIVITÀ

IL PROBLEMA È QUELLO DELL'ASSEGNAMENTO DI UNA RISORSA CONDIVISA DA UN CERTO NUMERO DI ATTIVITÀ IN COMPETIZIONE FRA LORO.

SIA $S=\{1,2,\ldots,n\}$ UN INSIEME DI n ATTIVITÀ CHE DEVONO UTILIZZARE UNA DETERMINATA RISORSA CHE NON PUÒ ESSERE UTILIZZATA CONTEMPORANEAMENTE.

UNA GENERICA ATTIVITÀ k È CARATTERIZZATA DA UN TEMPO DI INIZIO (ATTIVAZIONE) I_k E UN TEMPO DI FINE (CONCLUSIONE) F_k CON $I_k \leq F_k$.

DUE ATTIVITÀ k E j SONO DETTE *COMPATIBILI* SE GLI INTERVALLI $[I_k, F_k]$ E $[I_j, F_j]$ *NON SI SOVRAPPONGONO*.

IL PROBLEMA CHIEDE DI INDIVIDUARE UN INSIEME CHE CONTIENE IL MASSIMO NUMERO DI ATTIVITÀ MUTUAMENTE COMPATIBILI.

SI ASSUME CHE LE ATTIVITÀ IN INGRESSO SIANO ORDINATE IN MODO CRESCENTE RISPETTO AL LORO TEMPO DI FINE $F_1 \leq F_2 \leq F_3 \leq \ldots \leq F_n$

LO SPAZIO DI RICERCA PUÒ ESSERE COSÌ DEFINITO. SE 1,…,n SONO LE ATTIVITÀ RELATIVE A UNA ISTANZA i, ALLORA LO SPAZIO DI RICERCA È L'INSIEME DI TUTTI POSSIBILI SOTTOINSIEMI DI {1,…,n}.

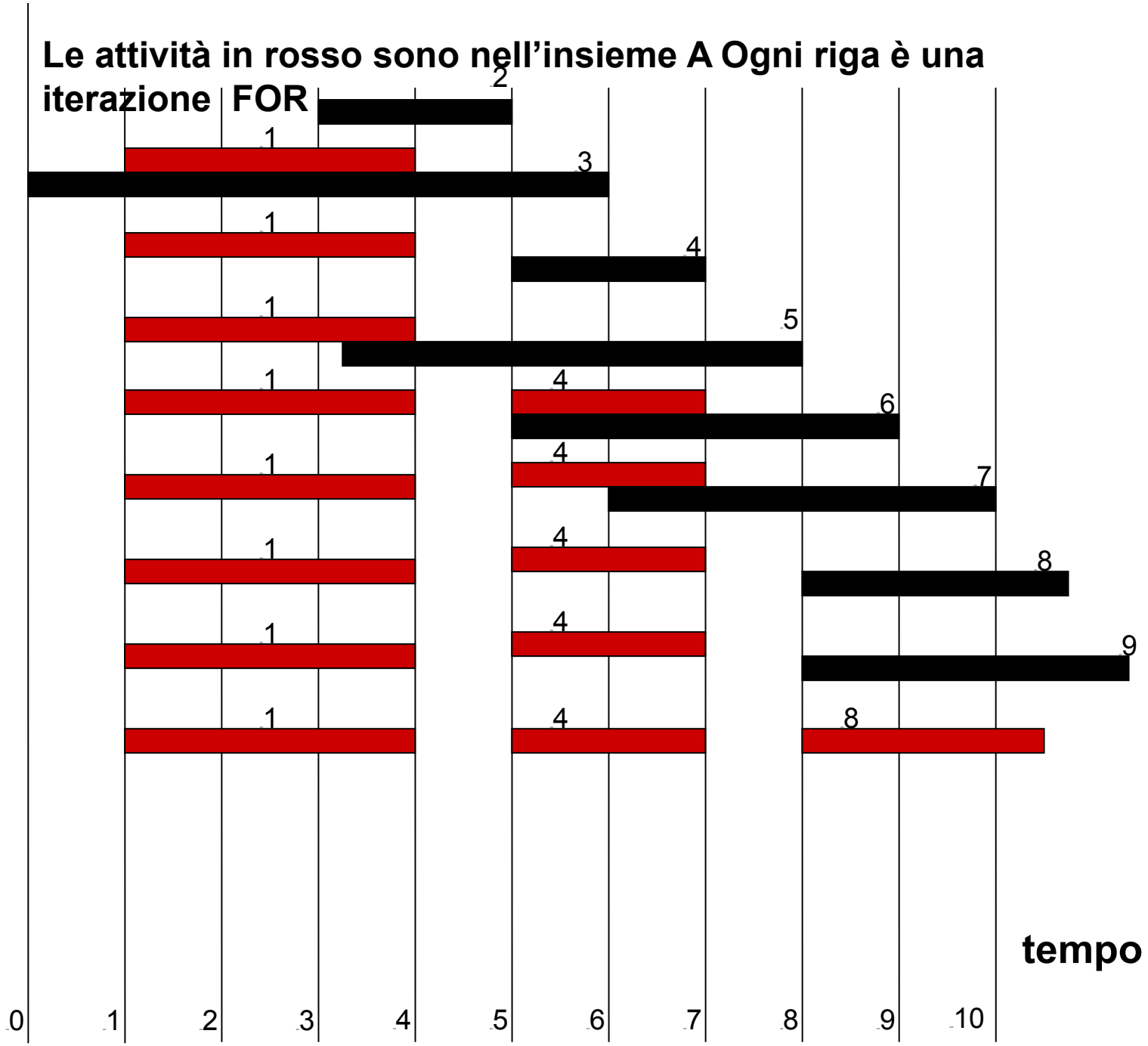
AD OGNI STADIO L'ALGORITMO GOLOSO SCEGLIE L'ATTIVITÀ k CHE, TRA QUELLE ANCORA DISPONIBILI, RILASCIA PER PRIMA LA RISORSA CONDIVISA (MINOR TEMPO F_k).

UNA VOLTA SCELTA L'ATTIVITÀ k, SI AGGIORNA L'INSIEME DI ATTIVITÀ DISPONIBILI, ELIMINANDO DALL'INSIEME QUELLE INCOMPATIBILI CON k, CIOÈ QUELLE CHE RICHIEDONO L'USO DELLA RISORSA NEL TEMPO CHE INTERCORRE TRA I_k E F_k

```
GREEDY (I e F: VETTORE; n: INTEGER)
Variabile A: INSIEME                    /*CONTIENE LE ATTIVITÀ*/

  CREAINSIEME (A)
  INSERISCI (1,A)
  j ← 1
  for k=2 to n do
    if Ik ≥ Fj then
      INSERISCI (k,A)
      j ← k
```

k	l_k	F_k
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14



ESEMPIO:

IL PROBLEMA DEL PERCORSO PIÙ BREVE IN UN GRAFO (RISOLTO CON UN ALGORITMO GENERATIVO)

SIA G=(N,A) UN GRAFO ORIENTATO ETICHETTATO NEGLI ARCHI CON VALORI INTERI POSITIVI.

TROVARE LA LUNGHEZZA DEL PERCORSO PIÙ BREVE CHE, DATO UN NODO $r \in N$, $\forall$ NODO $u \in N$, CONNETTA r AD u.

ESEMPIO: SE I NODI SONO CITTÀ E GLI ARCHI RAPPRESENTANO LE STRADE CHE LE COLLEGANO, TROVARE IL PERCORSO PIÙ BREVE TRA TUTTI QUELLI CHE CONNETTONO LA CITTÀ A ALLA CITTÀ B.

IL METODO

SI 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, SI 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** A **i** PASSANDO SOLO PER NODI CONTENUTI IN **S** (A PARTE i STESSO). L'IPOTESI DI FONDO E' 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 $i \notin S$.

INDIVIDUATO u SI INSERISCE IN S E SI AGGIORNA DIST PER I NODI CHE $\notin S$.

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 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 VALORE MAGGIORE DI TUTTE LE ALTRE ETICHETTE (MAXINT).

UNA **SOLUZIONE AMMISSIBILE T È OTTIMA SE E SOLO SE**

$$DIST(i) + C_{ij} = DIST(j) \;\forall\; (i,j) \in T \quad E$$

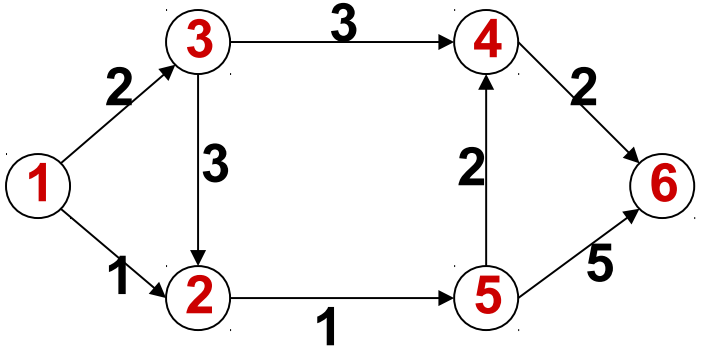
$$DIST(i) + C_{ij} \geq DIST(j) \;\forall\; ARCO(i,j) \in A$$

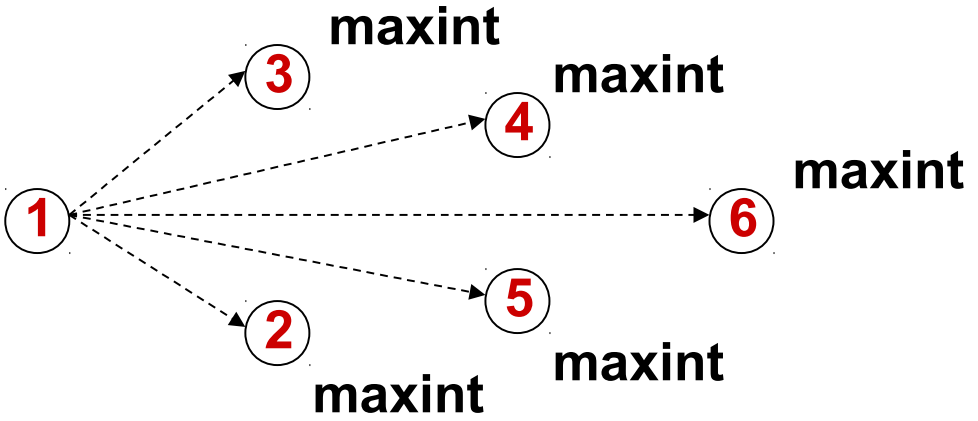
(CONDIZIONE DI BELLMAN)

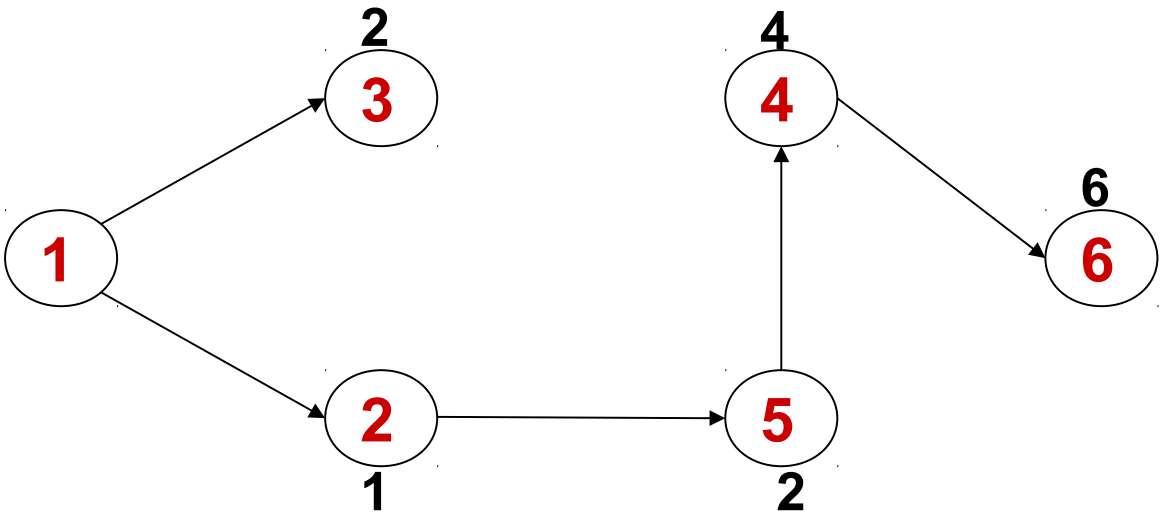
NOTA: ***UN ALTRO ALGORITMO (BELLMAN-FORD) RISOLVE IL PROBLEMA DEI CAMMINI MINIMI NEL CASO PIU' GENERALE IN CUI I PESI DEGLI ARCHI POSSONO ESSERE NEGATIVI***

```
CAMMINIMINIMI (G: GRAFO per riferimento; r:NODO)

  CREAINSIEME(S)
  T(r) ← 0, DIST(r) ← 0
  for k ← 1 to n do
    if k≠r then
      T(k) ← r
      DIST(k) ← MAXINT
    INSERISCI(r,S)
    while not INSIEMEVUOTO(S) do
      i ← LEGGI(S)
      CANCELLA(i,S)
      for j∈A(i) do /*A(i) È L'INSIEME DI ADIACENZA*/
        if DIST(i) + Cij < DIST(j) then
          T(j) ← i
          DIST(j) ← DIST(i) + Cij
          if not APPARTIENE(j,S) then
            INSERISCI (j,S)
```


3
4
2
2
3
2
1
6
1
2
5
5
1
2


maxint
3
maxint
4
maxint
1
6
5
maxint
2
maxint


2
3
4
4
6
1
6
2
5
1
2

ALGORITMO DI DIJKSTRA

SE LA STRUTTURA S E' UNA CODA CON PRIORITA', RAPPRESENTATA MEDIANTE UNA LISTA NON ORDINATA, SI OTTIENE UN ALGORITMO NOTO DAL 1959 E ATTRIBUITO A DIJKSTRA.
IN QUESTO CASO LE OPERAZIONI *LEGGI* E *CANCELLA* SONO GLI OPERATORI BASICI DISPONIBILI NELL'ALGEBRA DELLA CODA CON PRIORITA':

MIN: (PRIORICODA) → TIPOELEM

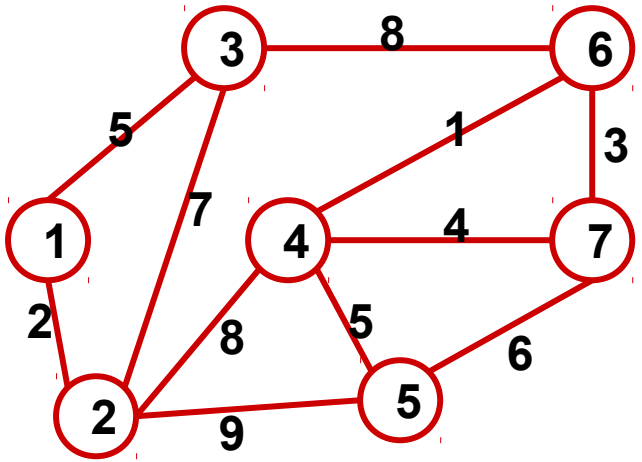
CANCELLAMIN: (PRIORICODA) → PRIORICODA

GLI ELEMENTI DELLA CODA CON PRIORITA' SONO I NODI DEL GRAFO E LE PRIORITA' ASSOCIATE ALTRO NON SONO CHE LE DISTANZE DAL NODO ORIGINE r. AD OGNI ITERAZIONE E' ESTRATTO DA S IL NODO AVENTE PRIORITA' (DISTANZA) MINIMA.

UN ALTRO PROBLEMA:

PROBLEMA DEL MINIMO ALBERO DI COPERTURA

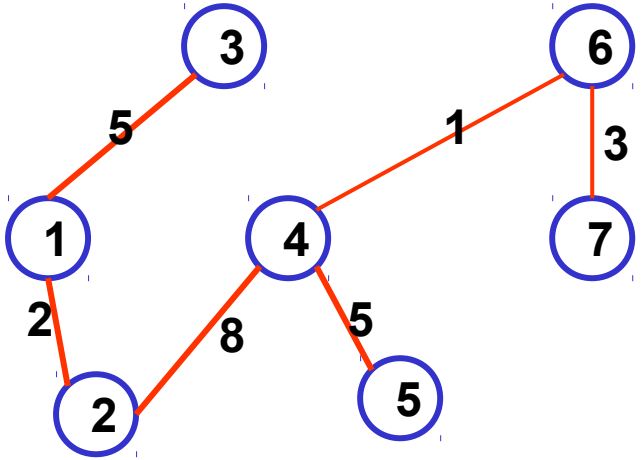
DATO UN GRAFO NON ORIENTATO E CONNESSO G=(N,A), CON PESI SUGLI ARCHI (NON NEGATIVI), TROVARE UN ALBERO DI COPERTURA PER G, CIOÈ UN ALBERO AVENTE TUTTI I NODI IN N, MA SOLO ALCUNI ARCHI IN A, IN MODO TALE CHE SIA MINIMA LA SOMMA DEI PESI ASSOCIATI AGLI ARCHI.



IL PROBLEMA PUÒ ESSERE RISOLTO CON MOLTI ALGORITMI, DEI QUALI I PIU' NOTI SI DEVONO A KRUSKAL (1956) E A PRIM (1957).

L'ALGORITMO DI KRUSKAL USA LA *TECNICA "GREEDY"*.

L'ALBERO DI COPERTURA MINIMO PER IL GRAFO PRECEDENTE E'



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 E' ESPRIMIBILE IN QUESTI TERMINI:

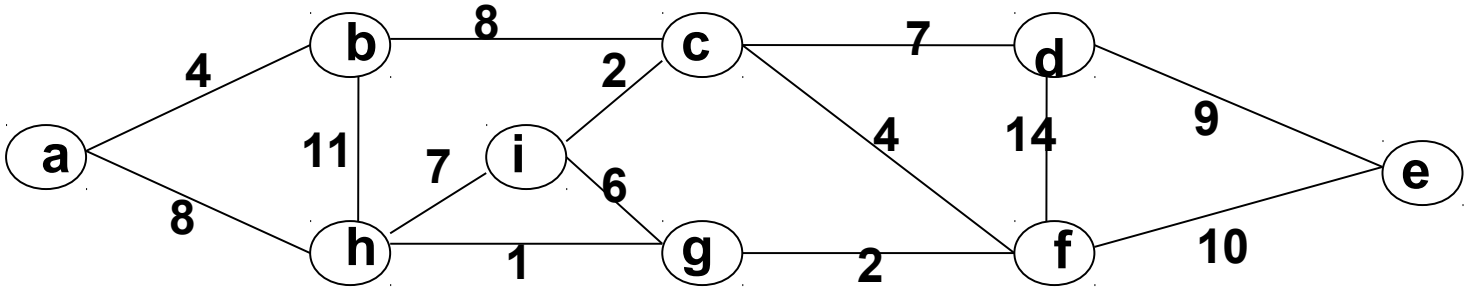
```
KRUSKAL(GRAFO)
  T ← Λ
  ORDINA GLI ARCHI DI G PER PESO CRESCENTE
  for a ← 1 to m do
    if (L'ARCO a=(i,j) NON FORMA CICLO
              CON ALTRI ARCHI DI T) then
      T ← T ∪ (a)
```

POSSIAMO RAPPRESENTARE IL GRAFO G COME PREFERIAMO. NELLA REALIZZAZIONE DATA DI SEGUITO IL GRAFO E' REALIZZATO CON UN VETTORE DI ARCHI E L'ALBERO T CON UNA LISTA DI ARCHI (REALIZZATA CON PUNTATORI).

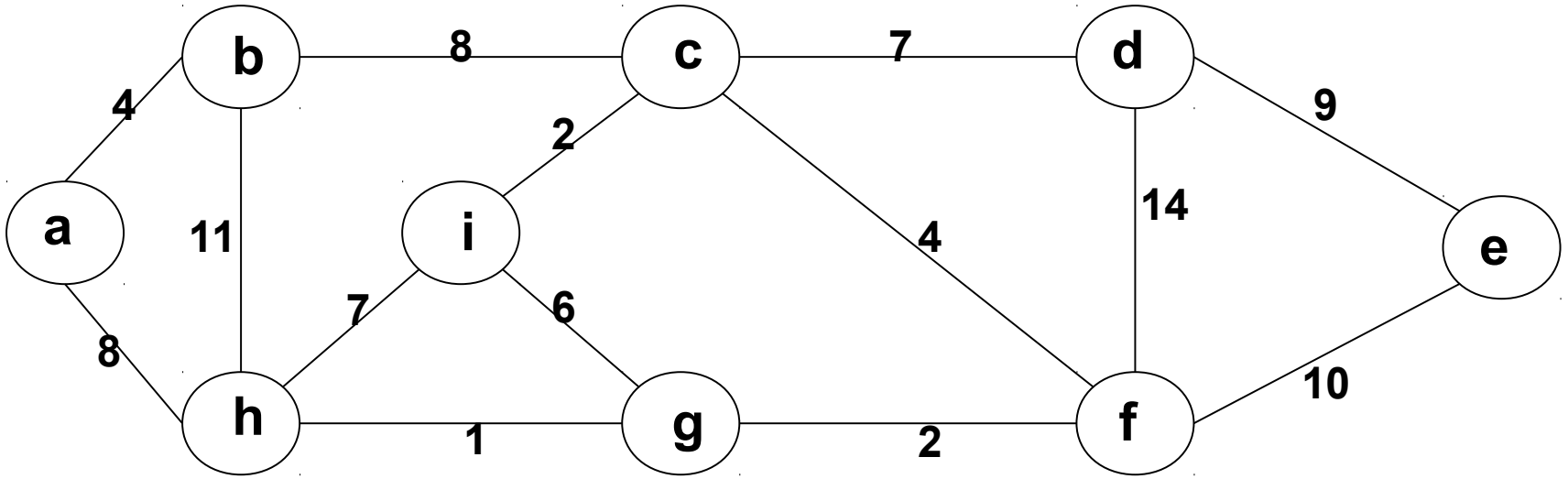
Definizione dei tipi:

ARCO: tipo strutturato con componenti
- i,j:INTEGER
- PESO: INTEGER

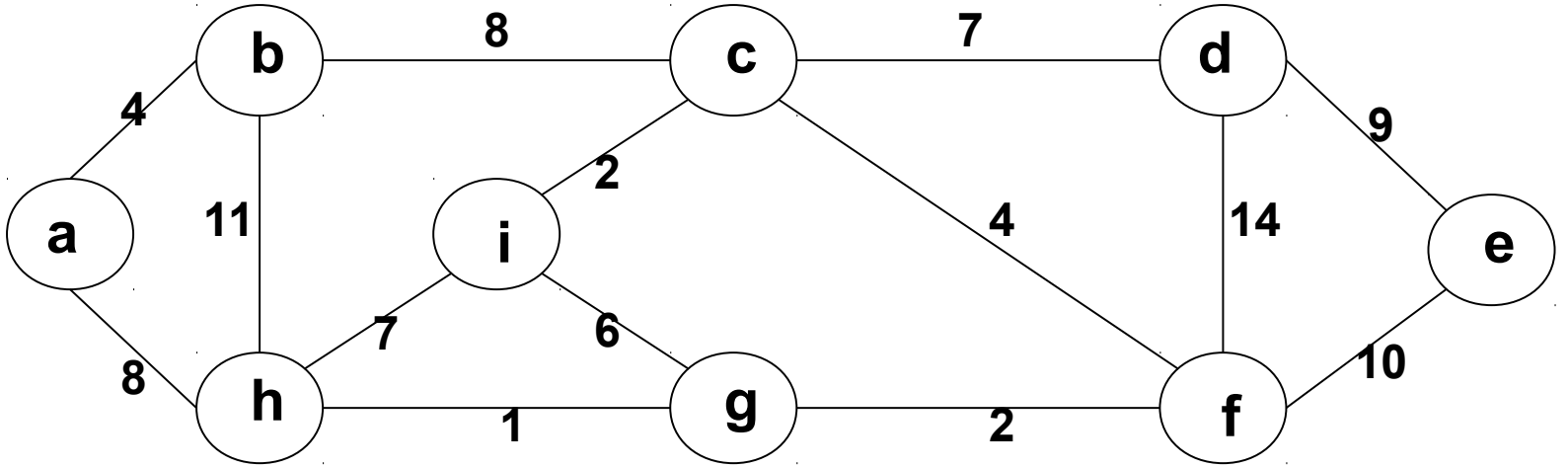
GRAFO: tipo strutturato con componenti
- A: ARRAY di MAXLUNG elementi di tipo ARCO
- n,m:INTEGER

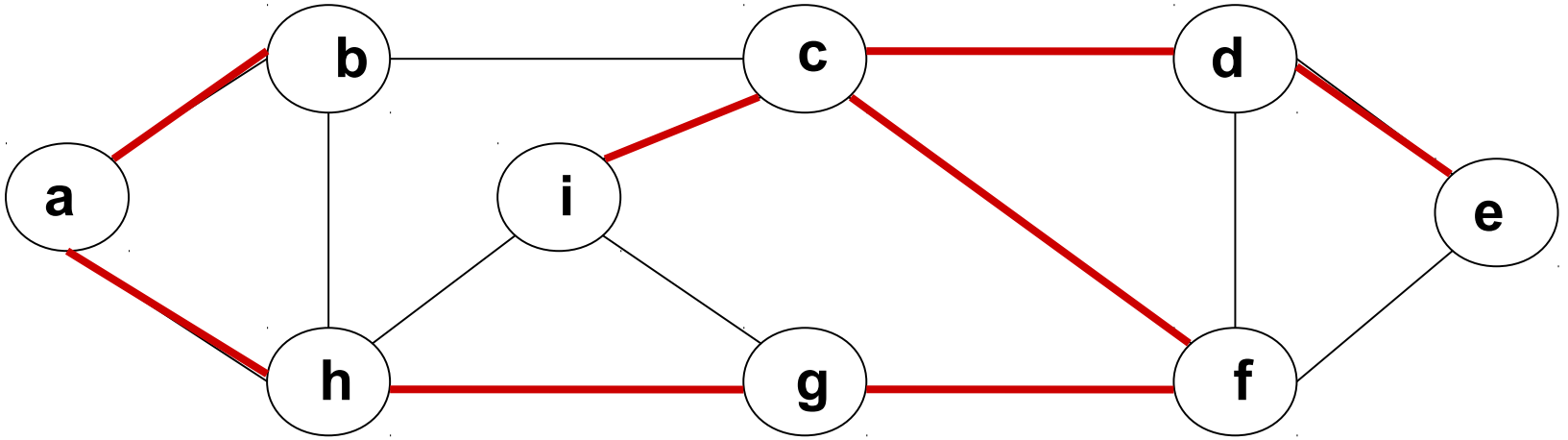


ECCO L'ORDINE IN CUI L'ALGORITMO CONSIDERA GLI ARCHI.



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 UNO.





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 E' NOTO, LA STRUTTURA MFSET (MERGE-FIND SET) È UNA PARTIZIONE DI UN INSIEME IN SOTTOINSIEMI DISGIUNTI DETTI COMPONENTI.

DUNQUE E' POSSIBILE UTILIZZARE UNA STRUTTURA DATI DI TIPO MFSET PER COSTRUIRE T.

```
KRUSKAL(G: GRAFO per riferimento)
Variabili:
  h: INTEGER;
  T: LISTA;
  S: MFSET; {PARTIZIONE DI UN INSIEME IN
                          SOTTOINSIEMI DISGIUNTI(COMPONENTI)}

                                                    CREALISTA(T)
  ORDINA G.A(1),…,G.A(G.m) PER ORDINE CRESCENTE DI G.A(h).PESO
  CREAMFSET(G.n,S)
  for h ← 1 to G.m do BEGIN
     if not TROVA (A(h).i,A(h).j,S) then
FONDI (A(h).i,A(h).j,S)
      INSLISTA (PRIMOLISTA(T),A(h),T)
```

LA TECNICA DIVIDE-ET-IMPERA

DERIVA DALL'IDEA DI DETERMINARE LA STRATEGIA DI UN PROBLEMA FACENDO RICORSO AL *PRINCIPIO DI DECOMPOSIZIONE INDUTTIVA*.

È NECESSARIO DISPORRE DI:

- **UNA RELAZIONE DI ORDINAMENTO SULLE ISTANZE DEL PROBLEMA, BASATA SULLA DIMENSIONE DELL'INPUT;**
- **UN METODO DI RISOLUZIONE DIRETTO PER TUTTE LE ISTANZE DEL PROBLEMA CHE NON SUPERANO UNA PREFISSATA DIMENSIONE LIMITE;**
- **UN MECCANISMO PER SUDDIVIDERE I DATI DI INGRESSO RELATIVI AD UNA ISTANZA IN DIVERSE PARTI, CIASCUNA DI DIMENSIONE MINORE DI QUELLA ORIGINARIA E RAPPRESENTANTE L'INPUT DI UNA NUOVA ISTANZA DELLO STESSO PROBLEMA;**
- **UN MECCANISMO PER COMPORRE LE SOLUZIONI PER LE ISTANZE INDIVIDUATE DALLA SUDDIVISIONE, PER OTTENERE LA SOLUZIONE PER L'ISTANZA ORIGINARIA.**

ALGORITMO DIVIDE-ET-IMPERA

1. **SE L'INPUT HA DIMENSIONE INFERIORE A UN CERTO VALORE k ALLORA UTILIZZA UN METODO DIRETTO PER OTTENERE IL RISULTATO**
2. **ALTRIMENTI, DIVIDI L'INPUT IN PARTI, CIASCUNA DI DIMENSIONE INFERIORE ALL'INPUT ORIGINARIO (*DIVIDE*)**
3. **ESEGUI RICORSIVAMENTE L'ALGORITMO SU CIASCUNO DEGLI INPUT INDIVIDUATI AL PASSO PRECEDENTE**
4. **COMPONI I RISULTATI OTTENUTI AL PASSO PRECEDENTE OTTENENDO IL RISULTATO PER L'ISTANZA ORIGINARIA (*IMPERA*)**

LA APPLICAZIONE PIU' NOTA DI QUESTA TECNICA SI HA NEGLI ALGORITMI DI ORDINAMENTO (NATURAL-MERGE-SORT E QUICKSORT)

ESEMPIO:

IL PROBLEMA DEL MINIMO E MASSIMO SIMULTANEI

IN ALCUNE APPLICAZIONI SERVE TROVARE IL MINIMO E IL MASSIMO IN UN INSIEME DI n ELEMENTI SIMULTANEAMENTE.

PER ESEMPIO, UN PROGRAMMA GRAFICO PUÒ AVER BISOGNO DI RAPPRESENTARE IN SCALA UN INSIEME DI DATI (x,y): IN QUESTO CASO VA DETERMINATO IL MINIMO E IL MASSIMO DI OGNI COORDINATA.

CERCANDO IL MINIMO E IL MASSIMO IN MODO INDIPENDENTE CI VORRÀ UN TOTALE DI 2(n–1) CONFRONTI.

MANTENENDO GLI ELEMENTI MINIMO E MASSIMO VIA VIA INCONTRATI E CONFRONTANDO I DUE ELEMENTI DELLA COPPIA IN INPUT SONO SUFFICIENTI 3(n/2) CONFRONTI.

L'ALGORITMO DEL MASSIMO E MINIMO SIMULTANEO:

1. SE LA DIMENSIONE DEL VETTORE NON SUPERA 2, ALLORA CALCOLA DIRETTAMENTE, MEDIANTE UN UNICO CONFRONTO, IL MINIMO E IL MASSIMO
2. ALTRIMENTI, DIVIDI IL VETTORE IN DUE SOTTOVETTORI DELLA STESSA DIMENSIONE, CALCOLA RICORSIVAMENTE IL MINIMO **MIN1** E IL MASSIMO **MAX1** DEL **PRIMO SOTTOVETTORE** E IL MINIMO **MIN2** E **MAX2** DEL **SECONDO SOTTOVETTORE**
3. DETERMINA IL MINIMO E IL MASSIMO DEL **VETTORE COMPLESSIVO CONFRONTANDO MIN1 CON MIN2 E MAX1 CON MAX2**

LA TECNICA DIVIDE-ET-IMPERA

È UNA TECNICA GENERATIVA CHE FA USO DELLA DECOMPOSIZIONE INDUTTIVA PER DETERMINARE IL METODO SOLUTIVO.

L'EFFICACIA DELLA TECNICA SI MANIFESTA ATTRAVERSO DUE ASPETTI:

- **CONSENTE DI PROGETTARE ALGORITMI SEMPLICI E INTUITIVI ATTRAVERSO L'INDUZIONE.**
- **SPESSO QUESTI ALGORITMI HANNO PRESTAZIONI MIGLIORI RISPETTO AD ALTRI (TIPICAMENTE A QUELLI DEL PARADIGMA SELETTIVO).**

L'EFFICIENZA DEGLI ALGORITMI DIPENDE DAL NUMERO a DI SOTTOPROBLEMI GENERATI, DALLA DIMENSIONE b DEI DATI IN INGRESSO AI SOTTOPROBLEMI E DAL COSTO f(n) NECESSARIO PER LA SCOMPOSIZIONE DEL PROBLEMA E LA COMPOSIZIONE DEI RISULTATI.