Ingegneria del Software
22a. Progettazione delle prove

Dipartimento di Informatica
Università di Pisa
A.A. 2014/15

prova (o collaudo o test)

- Verifiche (o validazioni) dinamiche
  - attività che prevedono l'esecuzione del software
  - in un ambiente controllato e con input e output definiti
  - sui moduli o sul sistema
- Metodo tanto intuitivo quanto complesso
  - pur essendo la forma di controllo più intuitiva (funzionerà? Beh, proviamo…) è la più complessa da realizzarsi (bene)
  - progettazione, esecuzione, analisi, debugging
  - validazione dei risultati, terminazione delle prove

caratteristiche

- Una prova non è sempre definitiva
  - è definita e ripetibile
  - i suoi risultati non sono estendibili: In linea di principio, i risultati di una prova valgono solo per le condizioni di quella prova
  - evidenzia un malfunzionamento (presenza di difetti): in altre parole (quelle di Dijkstra) la prova non potrà mai dimostrare l'assenza di difetti
- Le prove sono costose
  - occorrono molte risorse (tempo, uomini, macchine)
  - è necessario un processo definito
  - richiedono ulteriori attività di ricerca del difetto e correzione
  - il controllo dinamico è legato alle dimensioni dell'input, dell'output, dello stato e dell'ambiente operativo: variabili che, oltre ad essere grandi, sono spesso difficili da controllare e modellare

gli elementi di una prova

- Caso di prova (o test case)
  - una tripla <input, output, ambiente>
- Batteria di prove (o test suite)
  - un insieme (una sequenza) di casi di prova
  - una batteria può servire alla creazione di uno stato, alla copertura

- Procedura di prova
- le procedure (automatiche e non) per eseguire, registrare
  analizzare e valutare i risultati di una batteria di prove

conduzione di una prova

- Definizione dell'obiettivo della prova
  - è importante definire l'obiettivo
- Progettazione della prova
  - la progettazione consiste soprattutto nella scelta e nella definizione dei casi di prova (della batteria di prove)
- Realizzazione dell'ambiente di prova
  - ci sono driver e stub da realizzare, ambienti da controllare, strumenti per la registrazione dei dati da realizzare
- Esecuzione della prova
  - l'esecuzione può richiedere tempo
- Analisi dei risultati
  - l'esame dei risultati alla ricerca di evidenza di malfunzionamenti
- Valutazione della prova

progettazione

- Criteri funzionali
  - a scatola chiusa (black box)
  - basati sulla conoscenza delle funzionalità
  - mirati a evidenziare malfunzionamenti sospettati o comunque relativi a funzionalità identificate
- Criteri strutturali
  - a scatola aperta (white box)
  - basati sulla conoscenza del codice
  - mirati a esercitare il codice indipendentemente dalle funzionalità
- Gray box, una strategia più che un criterio (come vedremo in seguito)

statistico

- I casi di test sono selezionati in base alla distribuzione di probabilità dei dati di ingresso del programma
- Il test è quindi progettato per esercitare il programma sui valori di ingresso più probabili per il suo utilizzo a regime
- Il vantaggio è che, nota la distribuzione di probabilità, la generazione dei dati di test è facilmente automatizzabile
- Non sempre corrisponde alle effettive condizioni d'utilizzo del software
- È oneroso calcolare il risultato atteso

criteri funzionali

partizione dei dati d’ingresso

- Il dominio dei dati di ingresso è ripartito in classi di equivalenza
  - due valori d’ingresso appartengono alla stessa classe di equivalenza se, in base ai requisiti, dovrebbero produrre lo stesso comportamento del programma
- Il criterio è economicamente valido solo per quei programmi per cui il numero dei possibili comportamenti è sensibilmente inferiore alle possibili configurazioni d’ingresso
  - per come sono costruite le classi, i risultati attesi dal test sono noti e quindi non si pone il problema dell’oracolo
- Il criterio è basato su un’affermazione generalmente plausibile, ma non vera in assoluto
  - la deduzione che il corretto funzionamento sul valore rappresentante implichi la correttezza su tutta la classe di equivalenza dipende dalla realizzazione del programma e non è verificabile sulla base delle sole specifiche funzionali

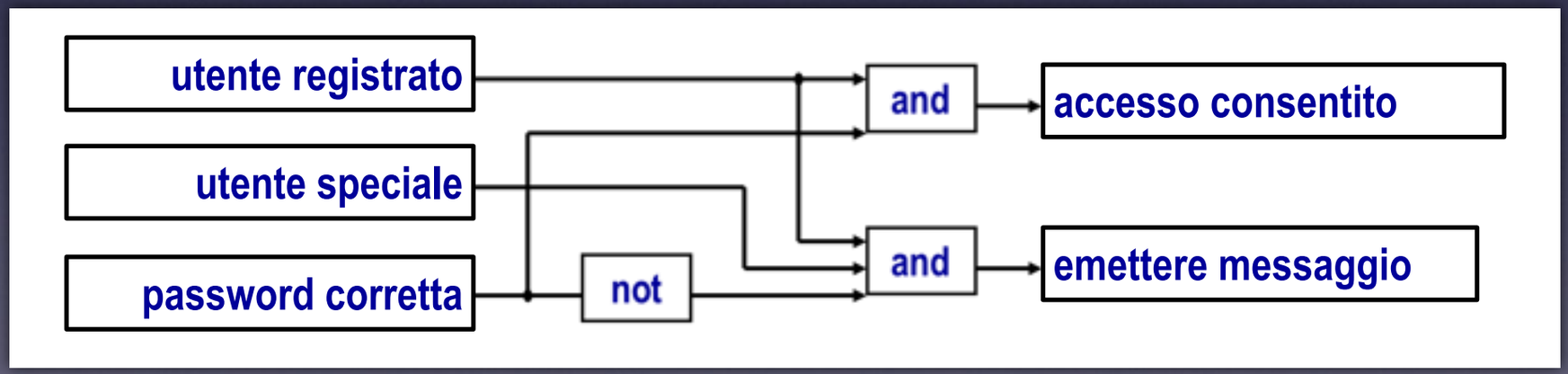
valori di frontiera

- Basato su una partizione dei dati di ingresso
    - le classi di equivalenza realizzate o in base all'eguaglianza del comportamento indotto sul programma o in base a considerazioni inerenti il tipo dei valori d'ingresso
- Dati di test: valori estremi di ogni classe di equivalenza

- È possibile che debba essere considerato il problema dell'oracolo
- Questo criterio richiama i controlli sui valori limite tradizionali in altre discipline ingegneristiche per le quali è vera la proprietà del comportamento continuo
    - in meccanica, ad esempio, una parte provata per un certo carico resiste con certezza a tutti i carichi inferiori
- Questa proprietà non è applicabile al software: i valori limite sono frequentemente trattati in modo particolare

un grafo causa-effetto

- Requisiti
  - l'accesso è consentito se l'utente è registrato e la password è corretta, è negato in ogni altro caso
  - se l'utente è speciale e la password è errata viene emesso un messaggio sulla console di sistema



- Grafo che lega un insieme di fatti elementari di ingresso (cause) e di uscita (effetti) in una rete combinatoria che definisce relazioni di causa-effetto

esercizio

- Dato un vettore di numeri d'esame che rappresenta esami, la funzione `numeroMedioEsami` ne restituisce la media, arrotondata all'intero superiore.

- Si forniscano cinque casi di prova per un test black box della funzione `numeroMedioEsami`, giustificando per ciascuno la ragion d'essere

soluzione

Casi di prova		Giustificazione
Input: valori	**Output: media**	
[ ]	0	Caso limite: vettore vuoto
[1]	1	Caso limite: un solo elemento
[1,1]	1	Caso speciale: tutti uguali
[1,2]	2	Verifica arrotondamento
[4,1,2]	3	Caso generico

- Solo interi positivi: si può assumere che il controllo relativo sia stato fatto al momento dell'inserimento

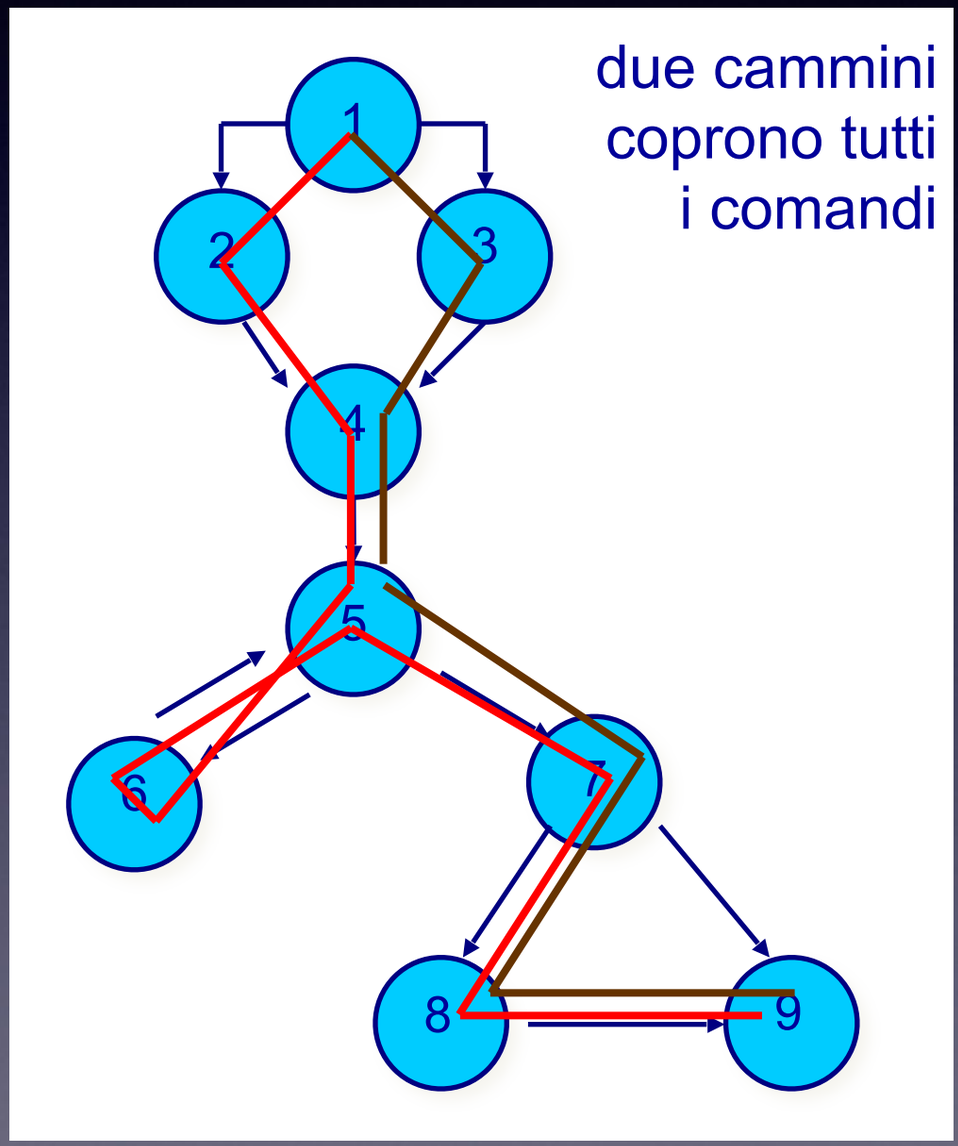
criteri strutturali

grafo di flusso

- Grafo di flusso
  - definisce la struttura del codice identificandone le parti
  - è ottenuto a partire dal codice
- I diagrammi a blocchi (detti anche diagrammi di flusso, *flow chart* in inglese) sono un linguaggio di modellazione grafico per rappresentare algoritmi (in senso lato)

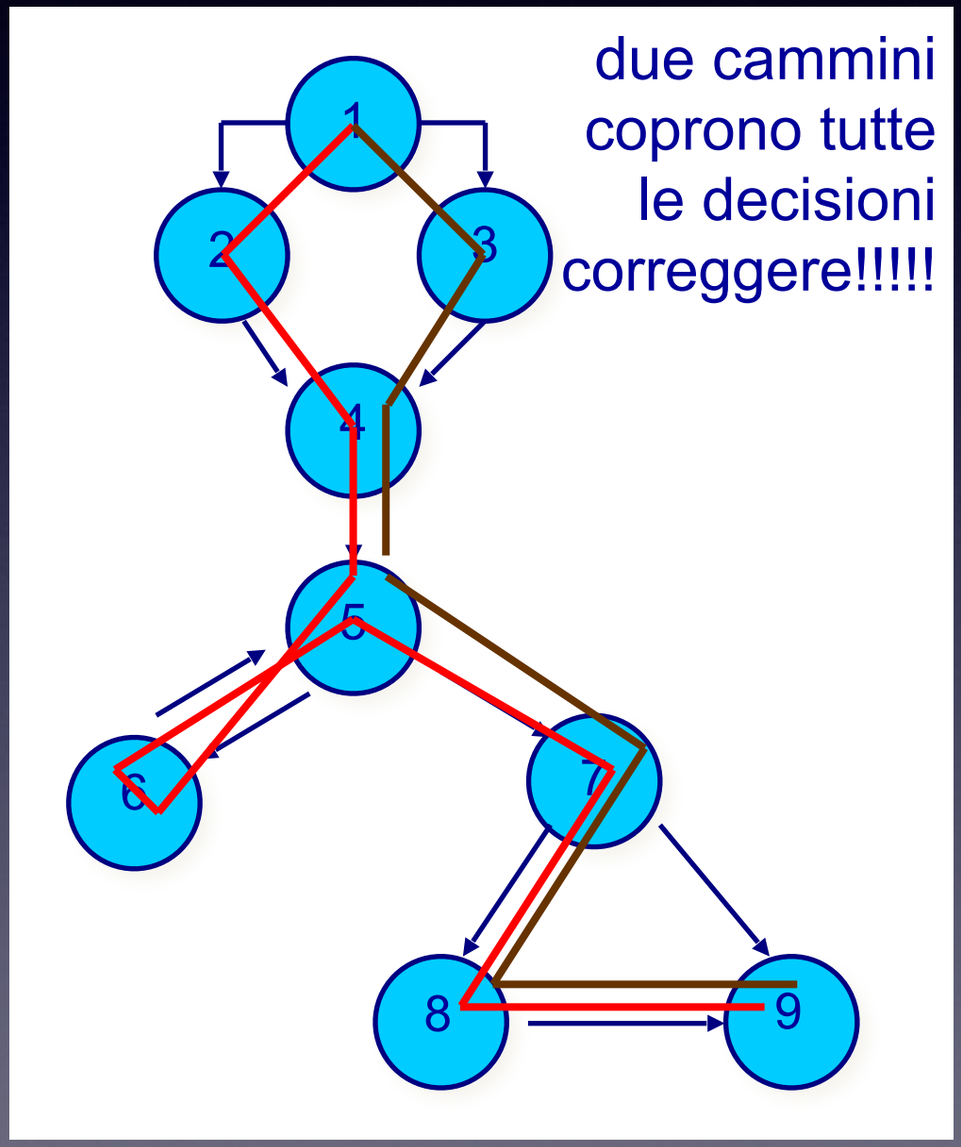
un grafo di flusso: comandi

```
double eleva(int x, int y) {
   1. if (y<0)
   2.    pow = 0-y;
   3.    else pow = y;
   4. z = 1.0;
   5. while (pow!=0)
   6.    { z = z*x; pow = pow-1 }
   7. if (y<0)
   8.    z = 1.0 / z;
   9. return z;
}
```



un grafo di flusso: decisioni

```
double eleva(int x, int y) {
   1. if (y<0)
   2.    pow = 0-y;
   3.    else pow = y;
   4. z = 1.0;
   5. while (pow!=0)
   6.    { z = z*x; pow = pow-1 }
   7. if (y<0)
   8.    z = 1.0 / z;
   9. return(z);
}
```



criteri di copertura (da *Binato et al.*)

- Statement coverage = copertura comandi (meglio: copertura istruzioni)
- Edge coverage = copertura degli archi = copertura delle decisioni
- Condition coverage = copertura delle condizioni (4 casi di test per un OR / AND)
  - si consideri il codice if (x>1 && y==0) {comando1} else {comando2}
  - e il test {x=2, y=0} e {x=2, y=1}
  - Il test garantisce la piena copertura delle decisioni, quindi dei rami e degli statement, ma non esercita tutte le combinazioni delle due condizioni in and
- Path coverage = copertura dei cammini: 4 casi per 2 if in sequenza
- Copertura cicli: si decide quanti

funzionali vs. strutturali

- Generalità degli approcci
  - rispetto alla validità dei risultati
  - rispetto alle caratteristiche da provare
  - rispetto ai costi da sostenere
- Dipendenze e implicazioni
  - l’applicazione dei criteri funzionali non dipende dal codice
  - i criteri strutturali si prestano alla valutazione della copertura

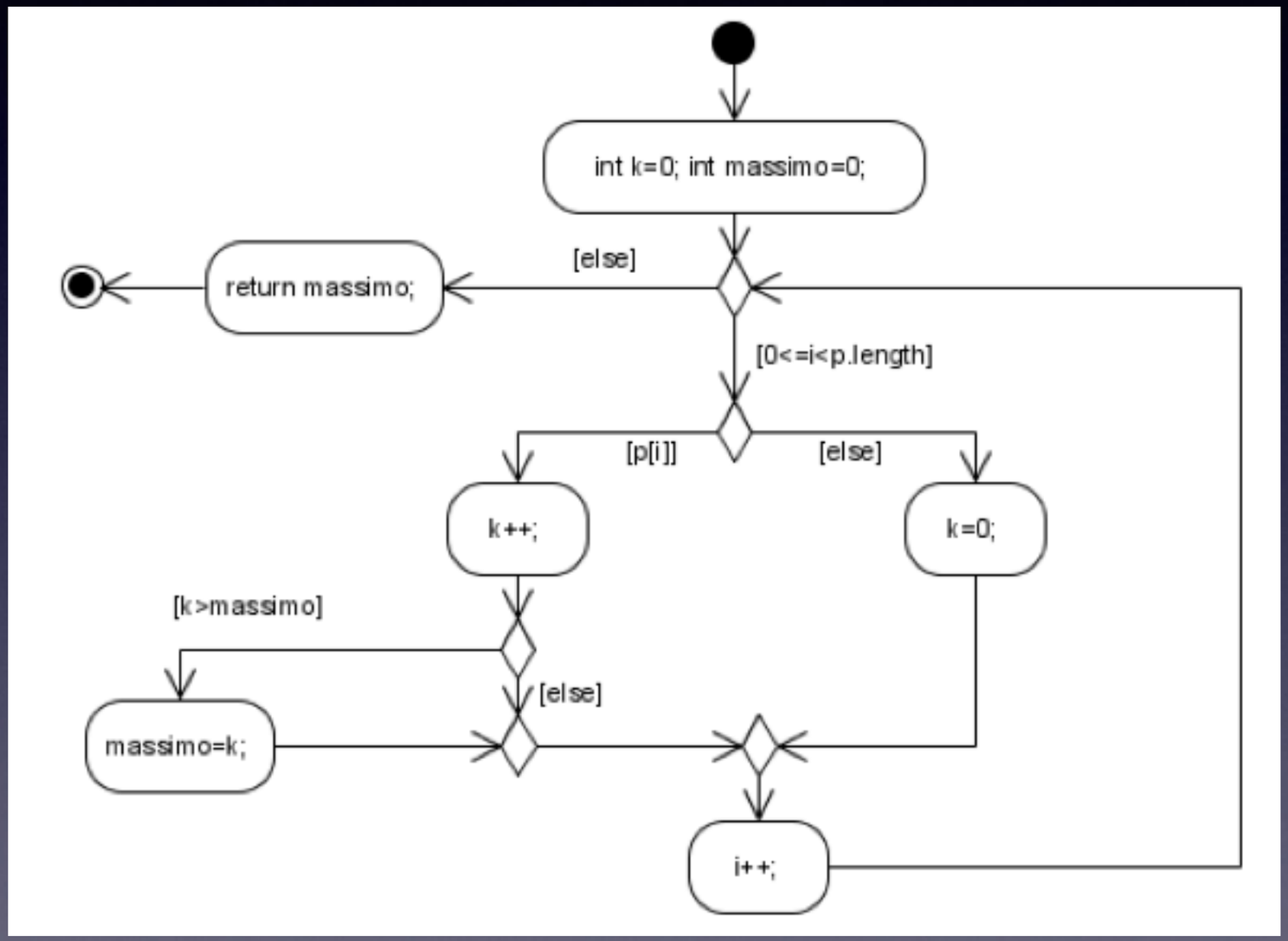
esercizio

- Il seguente metodo determina la durata del più lungo periodo di occupazione di una stanza

```
public int massimoPeriodo (boolean [] p) {
   int k = 0, massimo = 0;
   for (int i = 0; i < p.length; i++) {
      if (p[i]) { k++; if (k > massimo) massimo = k; }
      else k = 0;
   }
   return massimo;
}
```

soluzione

- Si disegni il grafo di flusso del metodo e si fornisca un insieme di cardinalità minima di casi di prova per la copertura delle decisioni



Per la copertura è sufficiente un vettore [t, f, t] (ma per riusarli nei testi funzionali si potevano provare anche casi limite tipo [], [f], …)