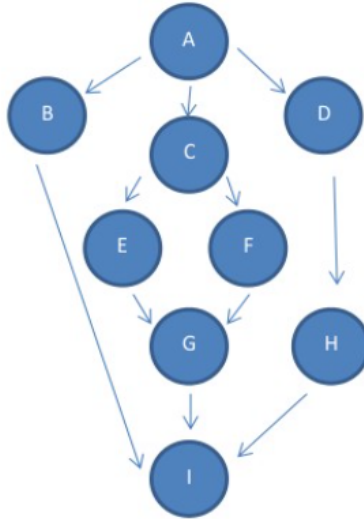


## Sistemi Operativi – Lab 11 – 13.01.20 - A.A. 2019/2020 - Prof. L. Sterpone

**Goal:** Implementazione e uso dei semafori.

### Esercizio 1 (tutorial semafori):

Implementare il grafo di precedenza illustrato nella figura seguente utilizzando i threads e il meccanismo di sincronizzazione basato sui semafori considerando i threads come non ciclici.



### Tutorial:

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

sem_t semB, semC, semD, semE, semF, semG, semH, semI;

void *tfA (void *arg) {
    sleep(1);
    printf("  A\n");
    sem_post(&semB);
    sem_post(&semC);
    sem_post(&semD);
}

void *tfB (void *arg) {
    sem_wait(&semB);
    sleep(2);
    printf("B  ");
    sem_post(&semI);
}

void *tfC (void *arg) {
    sem_wait(&semC);
    sleep(3);
    printf("C  ");
    sem_post(&semE);
    sem_post(&semF);
}

void *tfD (void *arg) {
    sem_wait(&semD);
    sleep(4);
    printf("D\n");
    sem_post(&semH);
}
```

```

void *tfE (void *arg) {
    sem_wait(&semE);
    sleep(1);
    printf(" E ");
    sem_post(&semG);
}

void *tfF (void *arg) {
    sem_wait(&semF);
    sleep(2);
    printf(" F\n");
    sem_post(&semG);
}

void *tfG (void *arg) {
    sem_wait(&semG);
    sem_wait(&semG);
    sleep(1);
    printf(" G");
    sem_post(&semI);
}

void *tfH (void *arg) {
    sem_wait(&semH);
    sleep(5);
    printf(" H\n");
    sem_post(&semI);
}

void *tfI (void *arg) {
    sem_wait(&semI);
    sem_wait(&semI);
    sem_wait(&semI);
    sleep(1);
    printf(" I\n");
}

int main (int argc, char **argv) {
    pthread_t thIDA, thIDB, thIDC, thIDD, thIDE, thIDF, thIDG, thIDH, thIDI;

    sem_init (&semB, 0, 0);
    sem_init (&semC, 0, 0);
    sem_init (&semD, 0, 0);
    sem_init (&semE, 0, 0);
    sem_init (&semF, 0, 0);
    sem_init (&semG, 0, 0);
    sem_init (&semH, 0, 0);
    sem_init (&semI, 0, 0);

    pthread_create (&thIDA, NULL, tfA, (void*)NULL);
    pthread_create (&thIDB, NULL, tfB, (void*)NULL);
    pthread_create (&thIDC, NULL, tfC, (void*)NULL);
    pthread_create (&thIDD, NULL, tfD, (void*)NULL);
    pthread_create (&thIDE, NULL, tfE, (void*)NULL);
    pthread_create (&thIDF, NULL, tfF, (void*)NULL);
    pthread_create (&thIDG, NULL, tfG, (void*)NULL);
    pthread_create (&thIDH, NULL, tfH, (void*)NULL);
    pthread_create (&thIDI, NULL, tfI, (void*)NULL);

    (void*)pthread_join(thIDA, (void*)NULL);
    (void*)pthread_join(thIDB, (void*)NULL);
    (void*)pthread_join(thIDC, (void*)NULL);
    (void*)pthread_join(thIDD, (void*)NULL);
    (void*)pthread_join(thIDE, (void*)NULL);
    (void*)pthread_join(thIDF, (void*)NULL);
    (void*)pthread_join(thIDG, (void*)NULL);
    (void*)pthread_join(thIDH, (void*)NULL);
    (void*)pthread_join(thIDI, (void*)NULL);

    return 0;
}

```

**Esercizio 2 (semafori):**

Implementare il grado di precedenza illustrato nella figura seguente utilizzando sia i threads (partendo dalla soluzione dell'esercizio precedente) che i processi (implementando i semafori con utilizzo delle pipe) considerando i thread e i processi come ciclici.

