Jolella – Condivisione Files in Rete Paritaria

Documentazione di Progetto per Laboratorio di Sistemi Operativi

Tutor: Stefano Pio Zingaro

A.A. 2018-2019

8 Maggio, 2019

Abstract

Il progetto propone la creazione un sistema distribuito per la condivisione di file decentralizzato, in una rete di entità pari tra loro. La realizzazione del progetto segue i principi della programmazione orientata ai servizi, rispettandone l'architettura e i paradigmi di gestione della concorrenza. I servizi e la comunicazione tra di essi sono implementati nel linguaggio di programmazione Jolie. Gli studenti che desiderano partecipare al progetto di questo anno accademico lavoreranno organizzati in gruppi composti da un minimo di quattro ad un massimo di cinque persone. La realizzazione del progetto mira a valutare la capacità degli studenti di lavorare in gruppo per lo sviluppo del codice, l'organizzazione del lavoro e la presentazione dei prodotti finali. Sarà richiesto ad ogni gruppo di apprendere le nozioni di base dei sistemi di *versioning* del codice per la creazione – collaborativa – di sorgenti software, ed, infine, di discutere le scelte implementative e le strategie utilizzate in sede di esame orale.

Indice

1 Informazioni Logistiche

In questa sezione sono riportate le istruzioni sulla formazione dei gruppi. Viene inoltre proposto un elenco di possibili date per la consegna dell'implementazione e del report. Tali informazioni sono soggette a cambiamenti ed a revisioni, ogni modifica viene comunicata attraverso la pagina web del tutor ed il forum ufficiale del corso. È possibile chiedere delucidazioni e prenotare un ricevimento via messaggio di posta elettronica, specificandone la motivazione, a questo indirizzo.

1.1 Formazione dei Gruppi

I gruppi sono costituiti da un minimo di quattro (4) ad un massimo di cinque persone (5), coloro che intendono partecipare all'esame comunicano entro e non oltre il **20 Maggio 2018** (pena esclusione dall'esame) la composizione del gruppo di lavoro, **via posta elettronica**, a questo indirizzo. Il messaggio ha come oggetto **GRUPPO LSO** e contiene:

1. Il nome del gruppo;
2. Una riga per ogni componente: cognome, nome e matricola;
3. Un indirizzo di posta elettronica di riferimento a cui mandare le notifiche, è incarico del proprietario trasmetterle agli altri membri.

Email di esempio con oggetto **GRUPPO LSO**

NomeGruppo

- *Vader Darth, 123456*
- *Pallino Pinco, 234567*
- *Pannocchia Anna, 345678*

Referente: anna.pannocchia@studio.unibo.it

Chi non riuscisse a trovare un gruppo invia allo stesso indirizzo di posta elettronica un messaggio con oggetto **CERCO GRUPPO LSO**, specificando:

1. Cognome, Nome, Matricola, Email;
2. Eventuali preferenze legate a luogo e tempi di lavoro (si cercherà di costituire gruppi di persone con luoghi e tempi di lavoro compatibili, nel limite delle possibilità del tutor).

Email di esempio con oggetto **CERCO GRUPPO LSO**

Sinalefe Pina, 234567, preferirei nei pressi del dipartimento, tutti i giorni dopo pranzo.

Le persone senza un gruppo vengono assegnate il prima possibile senza possibilità di ulteriori modifiche.

1.2 Date di Consegna ed Esame Orale

Le date a disposizione per la consegna dell'implementazione e del report, sono:

- le 23.59.59 di **Lunedí 1 Luglio** 2019;
- le 23.59.59 di **Lunedí 2 Settembre** 2019.

La data presa in considerazione per la consegna della parte di implementazione sarà quella di creazione del **Tag** su GitLab le istruzioni sulla consegna si trovano più avanti nella sezione 4. Solo in seguito alle consegne, vengono fissate data ed orario della discussione (notificate tramite la mail di riferimento), compatibilmente coi tempi di correzione. La discussione dell'implementazione e la relativa demo di funzionamento viene effettuata in un incontro unico con tutti i componenti presenti. Al termine della discussione, ad ogni singolo componente verrà assegnato un voto in base all'effettivo contributo dimostrato nel lavoro. La valutazione è indipendente dal numero di persone che compongono il gruppo.

2 Descrizione delle Componenti del Sistema

In questa sezione vengono descritte le componenti del sistema preso in considerazione per la realizzazione del progetto Jolella. Tali descrizioni hanno il solo scopo di informare, e non di porre vincoli sulle scelte implementative. Alcuni di questi vincoli e limitazioni sono invece discussi nella sezione 3, insieme alle motivazioni delle stesse.

2.1 La rete Peer-To-Peer

Il termine peer-to-peer [1, 2] si riferisce ad una classe di sistemi ed applicazioni che utilizzano risorse distribuite per eseguire una funzionalità (critica) in modo decentralizzato [3].

Ogni nodo in una rete P2P viene chiamato generalmente **peer**, ha la medesima importanza e svolge le stesse funzioni di tutti gli altri nodi.

Le caratteristiche di un sistema P2P possono essere riassunte nelle seguenti:

- i peer sono indipendenti (autonomi);
- ogni peer ha funzioni di **client** e di **server** contemporaneamente, e condivide delle risorse (in questo caso files);
- possono essere presenti dei nodi con funzionalità diverse rispetto agli altri, ad esempio per il monitoraggio o, in generale, per offrire operazioni specifiche;

- il sistema è **altamente distribuito**, il numero di peers può essere dell'ordine delle centinaia di migliaia;
- il sistema è **altamente dinamico**, un peer può entrare ed uscire dalla rete in ogni momento.

  N.B.

  - le operazioni di ingresso/uscita (**join**/**leave**) dalla rete fanno parte dell'interfaccia di ogni peer.

2.1.1 Architetture di una rete P2P

Esistono tre tipologie di reti distribuite peer-to-peer:

1. Decentralizzate pure – dove tutti i nodi sono peer, non esiste nessun coordinatore centralizzato. Ogni peer può funzionare come client e/o server.
2. Parzialmente centralizzate – in queste reti alcuni nodi (supernodi) facilitano l'interconnessione tra i peer, questi hanno, di solito, un indice locale centralizzato dei peer locali. Il pattern di comunicazione è: un peer comunica con il supernodo, che a sua volta comunica con gli altri peer.
3. Decentralizzate ibride – queste reti presentano un server centralizzato che facilita l'interazione tra i peer (servizio di **localizzazione**, ad esempio tramite **location**).

Una rete P2P per la condivisione di files. La condivisione di files costituisce il principale utilizzo delle reti decentralizzate fin dagli anni '90 (si parla di quasi il 30% della rete Internet). Perchè una rete P2P sia abilitata alla condivisione di files è necessario che ogni nodo esponga delle funzionalità di **pubblicazione**, **ricerca** e **recupero**.

Vediamo un "classico" esempio di comunicazione tra due entità in una rete peer-to-peer per la condivisione di un file:

1. Alice esegue un **Client** P2P;
2. Alice registra il suo contenuto nel sistema P2P;
3. Alice cerca "Hey Jude";
4. Il **Client** visualizza altri peer che hanno una copia di "Hey Jude";
5. Alice sceglie uno, o più peer, per esempio Bob;
6. il file viene copiato da Bob ad Alice;
7. mentre Alice esegue il download, altri peer chiedono ad Alice l'upload di file.

L'esempio riportato, senza pretesa di descrivere in modo completo il flusso di comunicazioni, evidenzia alcune importanti proprietà che un **Client** di una rete P2P per la condivisione files deve soddisfare. Esso, infatti, non solo consente ad Alice di registrare un file per la sua condivisione ma, "sotto il tappeto", si occupa, per esempio, di fare il *join* della rete prima di tutto il resto. Inoltre, permette ad Alice di copiare files condivisi da altri utenti della rete e consente di individuare quali peer possiedono un determinato file, tramite una ricerca basata su **keyword**.

Quest'ultimo problema, cioè quello della ricerca in una rete decentralizzata, può essere ricondotto a quello del *routing* o instradamento [1]. Nei sistemi P2P per la condivisione dei files il routing dipende dall'architettura del sistema stesso, nel caso di sistemi decentralizzati ibridi, questo si effettua mediante l'uso di una **directory centralizzata** [2]. Nel caso invece di architetture decentralizzata pura, la ricerca utilizza il protocollo di **flooding**.

Sistemi con directory centralizzata. In questi sistemi un nodo centralizzato, chiamato anche **directory server** possiede una mappa con tutte le risorse dei peer, in un modo molto simile a quello che usano i sistemi di ricerca dei computer che usiamo tutti i giorni (*Siri*, *Cortana*, etc.). Il servizio del (super-)nodo fornisce operazioni di **discovery** dei peer e di **lookup** delle risorse.

Tra le criticità più importanti di questa tecnica esiste il costo di gestione del server centralizzato, in termini di risorse, la scalabilità limitata, dovuta all'approccio centralizzato (problemi di collo di bottiglia vengono spesso riscontrati in questi sitemi) e, di conseguenza, la presenza di un *single point of failure* nel sistema.

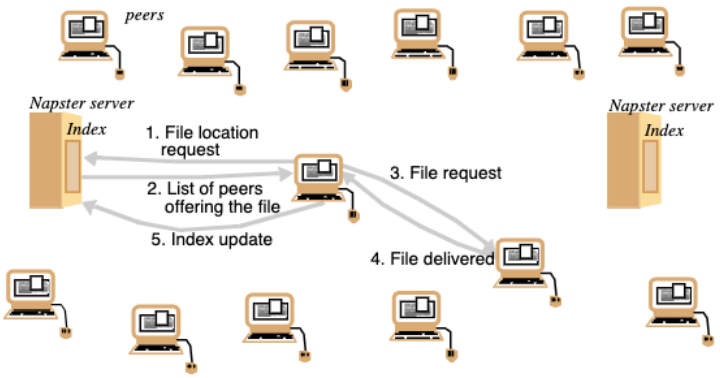

Figure 1: Rappresentazione grafica di un esempio di sistema con directory centralizzata. Fonte Uniroma.

[1] Per una veloce lettura su questo argomento è possibile consultare la sezione dell'articolo relativo su Wikipedia: https://it.wikipedia.org/wiki/Protocollo_di_routing.

[2] Un esempio di programma di condivisione files basato su questa tecnica è il famoso Napster.

Sistemi con ricerca flood-based. Costituiscono l'approccio più decentralizzato, in cui ogni peer propaga – da qui il termine inglese *flood* – la richiesta ai peer intorno a lui, che a loro volta inviano la richiesta ai peer "vicini", escludendo quello da cui hanno ricevuto la richiesta. La risposta, una volta trovata, viene inviata direttamente al peer originario della richiesta. I sistemi così configurati devono porre particolare attenzione nella gestione della propagazione della richiesta, in quanto, senza un'opportuna strategia di *back-off*, essa potrebbe circolare all'infinito. Una delle tecniche più usate è quella che utilizza un *TTL* (Time To Live), un intero contenuto in ogni query inviata. Ad ogni passaggio per un nodo della rete, il peer decrementa questo TTL che, arrivato a 0, invalida la richiesta e ferma la propagazione [3].

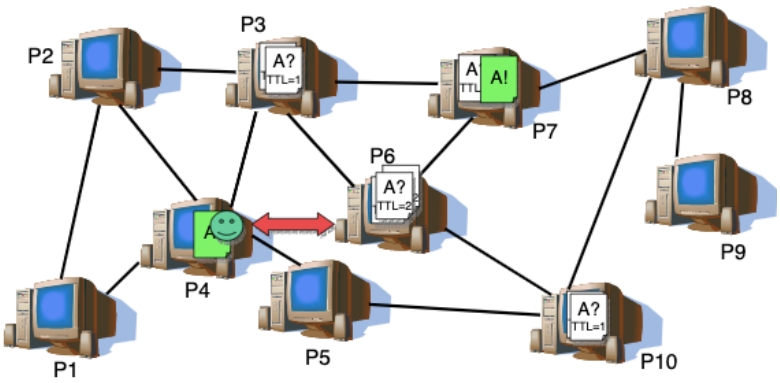

Figure 2: Rappresentazione grafica di un esempio di sistema con directory centralizzata. Fonte Uniroma.

Note criticità del flooding riguardano la congestione della rete per il numero di messaggi scambiati da sistemi in cui i messaggi di richiesta cercano di esplorare l'intero insieme dei nodi. A questo è collegato anche il problema della determinazione del raggio di ampiezza del flooding (e quindi il TTL). Non esiste quindi garanzia che venga interrogato il nodo che possiede la risorsa pur sapendo che questa esiste nella rete. Un'altra limitazione di questi sistemi risiede nella nozione di "vicinanza" dei nodi, non è chiaro infatto come definire questo parametro, è spesso necessario effettuare dei tentativi per stabilire quale sia il valore migliore per un determinato sistema.

3 Discussione sulle Strategie di Implementazione

Nella presente sezione vengono discusse le scelte implementative per la realizzazione di un sistema come quello descritto nella sezione 2. Le componenti

[3] Anche in questo caso l'articolo di Wikipedia mostra, brevemente, alcuni concetti chiave della tecnica di flooding e del TTL, anche detto *Hop Count* https://en.wikipedia.org/wiki/Flooding_(computer_networking).

vengono infatti declinate nel caso specifico di una rete (Jolella) che presenterà differenze (anche se piccole) per ogni gruppo di studenti coinvolto nella realizzazione di questo progetto.

3.1 La rete Jolella

Jolella, ispirata al programma Gnutella [4], è una rete decentralizzata per la condivisione di files tra nodi della rete stessa (anche detta *P2P*) [4].

Architettura di Jolella. La scelta dell'architettura della rete non è vincolata, il progetto potrà infatti essere svolto seguendo una delle tre strategie proposte. Viene invece richiesta coerenza nelle scelte implementative derivanti dall'architettura selezionata, ed una concisa documentazione sulle motivazioni che hanno portato a tale scelta.

3.1.1 Obiettivi e Problemi dei sistemi P2P

Esistono motivazioni specifiche per cui si è scelto il sistema P2P nella realizzazione del progetto proposto. Alcune risiedono nelle caratteristiche del sistema stesso, altre, legate ad esigenze didattiche, mirano a mettere in luce il livello di apprendimento raggiunto su alcuni concetti teorici specifici (ad esempio quelli relativi alla **gestione della concorrenza**).

Tra le caratteristiche del sistema alcune sono interessanti rispetto alle soluzioni "centralizzate". Queste sono l'aumento di scalabilità (la creazione di un peer e del milionesimo peer è uguale); l'interoperabilità e la possibilità di aggregazione delle risorse; l'autonomia ed il dinamismo (se un peer si disconette, la rete non subisce modifiche).

Alcune limitazioni sono imposte sulla strategia di realizzazione del progetto per innescare l'esigenza di risoluzione del problema proposto e consolidare così i concetti teorici. Le criticità dei sistemi P2P da considerare, sono (in ordine sparso): la **sicurezza e consistenza delle comunicazioni**, la **disuguglianza tra nodi** e **l'elevato dinamismo o instabilità della rete**. Una particolare attenzione va riservata a questi aspetti, la cui risoluzione costituisce il fulcro stesso del progetto.

Altri vincoli e limitazioni. Di seguito alcuni vincoli che, per semplicità, vengono imposti sul sistema:

- assumiamo che la ricerca per keyword, sia effettuata sui nomi dei file, e che questi corrispondano con l'effettivo contenuto (non sono *fake files*).
- assumiamo che la rete sia una rete **non strutturata**, cioè organizzata come un **grafo random** e dove non esistono vincoli sul posizionamento dei nodi rispetto alla topologia del grafo. In questa configurazione l'aggiunta

[4] Creato nel 2000, è stato uno dei primi protocolli di condivisione di files su una rete decentralizzata peer-to-peer.

o la rimozione del nodo non comporta la riorganizzazione della rete. Per delucidazioni sui grafi consiglio i lucidi presenti in [5], alla voce “Grafi”.

- in caso di sistemi ad architettura puramente decentralizzata, e quindi con ricerca flood-based, si richiede l’utilizzo di un ID per ogni query, così da controllare che questa non sia già stata risolta dal nodo ricevente.

Non esistono vincoli di alcun tipo sull’uso di linguaggi (supportati da Jolie) per la creazione di servizi. L’unica limitazione è sull’uso di Jolie per implemetare la comunicazione tra i servizi creati.

Un esempio di servizio creato in *Java* (attualmente le tecnologie supportate da Jolie sono Java e Javascript) può essere trovato in questa *repository* di GitHub.

3.2 Il Jeer

Un peer della rete Jolella (che, per disambiguità, chiameremo **Jeer**) si può occupare sia delle operazioni di gestione della rete, sia delle funzionalità del singolo peer (descritte entrambe nella sezione 2.1).

In Gnutella, i peer offrono una operazione di **discovery** che permette ad un nodo che conosce l’indirizzo di un altro nodo, di accedere alle liste di nodi connessi. Per la ricerca, il nodo utilizza un algoritmo flood-based di tipo *breadth-first* (BFS).

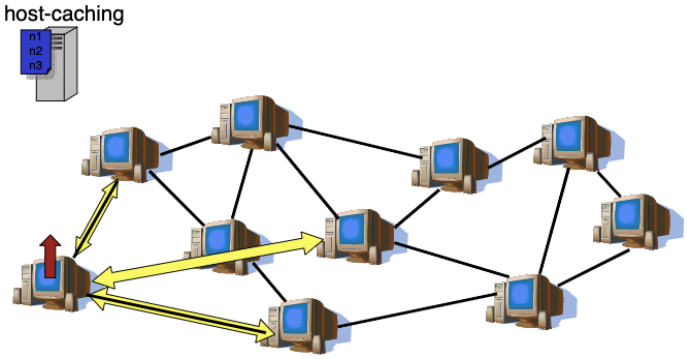

Figure 3: Un peer può rifiutare una richiesta di connessione, ad esempio perché ha raggiunto un numero massimo di connessioni ammesse.

3.3 Monitorare Jolella ed i Jeer

Abbiamo bisogno di un sistema di feedback per il monitoraggio della rete, sia in fase di *debugging*, che durante la manutenzione del sistema. Inoltre, durante la prova orale, sarò necessario per il controllo della corretta implementazione del soluzioni proposte dal gruppo. Il monitor può essere visto come una console di *logging*, dove è possibile visualizzare lo storico della rete. Esso può essere

stampato a video oppure su un file. Un esempio di output del monitor della rete potrebbe consistere nel semplice elenco numerato riportato in basso, **la numerazione è obbligatoria**.

```
1. La rete Jolella è attiva!
2. Il Jeer <id_jeer> ora partecipa alla rete.
3. Il Jeer <id_jeer> ha pubblicato il Jile <id_jile>
```

4 Specifiche per la Consegna

Globalmente, vengono consegnati due prodotti: la documentazione e il codice sorgente del progetto. La valutazione finale avviene mediante una discussione di gruppo (orale), nella quale vengono discusse le strategie con la quale i prodotti consegnati sono stati generati. Le domande possono essere rivolte a chiunque dei partecipanti al gruppo.

N.B.

1. non si accettano richieste di eccezioni sui progetti con motivazioni legate a esigenze di laurearsi o di non voler pagare le tasse per un altro anno.
2. chi copia o fa copiare, anche solo in parte, si vede invalidare completamente il progetto senza possibilità di appello. Il codice viene controllato con un programma per il rilevamento di plagio.

4.1 La documentazione

È possibile scrivere la documentazione nel formato preferito, l'importante è che il PDF generato rispetti la struttura del modello (riportato in basso 4.1). La documentazione ha lunghezza di quattro o cinque pagine (quindi da 8 a 10 facciate), è scritto con font di grandezza **12pt** e viene consegnato in formato PDF. Di seguito viene riportato un esempio di documentazione con le principali caratteristiche da inserire.

L'intestazione della Documentazione

- Jolella – Laboratorio Sistemi Operativi A.A. 2018-2019
- Nome del Gruppo
- Indirizzo mail di riferimento: nome.cognome@studio.unibo.it
- Componenti:
  - Cognome, Nome, Matr. 0000424242
  - ...

Il corpo della Documentazione

1. Descrizione generale del progetto – descrizione delle features implementate e del contenuto della documentazione.
2. Istruzioni per la demo – le istruzioni per eseguire una demo.
3. Discussione sulle strategie di implementazione:
   (a) Struttura del progetto – come è stato diviso il progetto, perché, i problemi principali riscontrati, le alternative considerate e le soluzioni scelte.
   (b) Sezione di descrizione della feature x – abbiamo implementato la funzione di 'foo' …(con esempi di codice).

4.1.1 Griglia di Valutazione

La valutazione della documentazione verte sull'analisi dello scritto e sulla sua capacità di esprimere con chiarezza i concetti descritti, soprattutto **grazie all'uso di esempi**.

In particolare la griglia di valutazione usata è la seguente:

Qualità dell'informazione	Riconoscimento dei problemi (di concorrenza) e loro descrizione.
Uso degli esempi	Presenza di almeno un esempio in tutte le scelte implementative.
Analisi delle scelte implementative	Descrizione della propria scelta implementativa e presenza di proposte di alternative valide.

4.2 Il codice sorgente

Il progetto viene sviluppato utilizzando il linguaggio Jolie. Non ci sono requisiti riguardo ai protocolli (*protocol*) e i media (*location*) utilizzati per realizzare la comunicazione tra i componenti del sistema.

La gestione del progetto avviene col supporto del sistema *git*, a questa pagina è possibile trovare una lista di comandi utili da tenere sempre a mente.

Il codice del progetto è contenuto in un repository sul server in cloud del servizio online GitLab e viene gestito seguendo la procedura qui descritta.

GitLab

- Ogni membro del gruppo crea un account su GitLab;
- il referente del gruppo crea un nuovo progetto cliccando sul + in alto a destra nella schermata principale di GitLab, inserendo il nome "LabSO_NomeGruppo" e cliccando su **New Project**;
- una volta che il progetto è stato creato, il referente aggiunge ogni membro del gruppo come **role permission >Developer** al progetto andando su **Settings > Members** nel menù a sinistra, cercandoli in base allo username registrato su GitLab;
- il referente aggiunge l'utente "stefanopiozingaro" come **role permission > Reporter**.

4.2.1 La consegna dell'implementazione tramite il Tag di GitLab

Al momento della consegna, il repository dovrà contenere i sorgenti del progetto e la relazione, nominata **REPORT_LSO.pdf**. Per effettuare la consegna:

1. nella pagina di GitLab del repository, cliccare sulle voci del menù **Repository > Tags > New Tag**;
2. digitare come **Tag Name** il nome **Consegna**;
3. cliccare su **Create Tag** per eseguire la creazione del **Tag** di consegna.

Una volta creato il Tag, inviare una email di notifica di consegna con soggetto **CONSEGNA LSO - NOME GRUPPO** a questo indirizzo di posta elettronica.

N.B.

Inoltre, La documentazione va consegnata in forma cartacea nella casella del Professor Sangiorgi al piano terra del Dipartimento di Informatica.

4.2.2 Griglia di Valutazione

La valutazione dell'implementazione del sistema si basa sull'analisi del codice Jolie, sull'uso dei costrutti del linguaggio per la creazione di soluzioni efficienti, sulla tolleranza ai guasti del sistema implementato e sulla gestione degli errori. In particolare la griglia di valutazione usata è la seguente:

4.3 Il test del progetto

Insieme alla documentazione ed al codice sorgente, dovrà essere preparato uno script che permette di automatizzare i test. Almeno nella fase iniziale della prova

Uso dei costrutti di Jolie	Corretto utilizzo dei costrutti per la gestione della concorrenza, uso corretto di `execution(…)`.
Distribuzione del carico di lavoro nel gruppo	Omogeneità nella ripartizione dei compiti nel gruppo, ogni membro partecipa egualmente allo sviluppo, indica il singolo contributo assenza di dissimmetria di informazione.
Grado di partecipazione alla comunità	Presenza di domande e risposte sul forum del corso

orale, può essere utile preparare una serie di *screenshot* correlati allo script, che permetteranno di velocizzare le operazioni di controllo del codice. Tale suite si test può essere intergrata nel codice sorgente

References

[1] Androutsellis-Theotokis, S., and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv. 36*, 4 (Dec. 2004), 335–371.

[2] Balakrishnan, H., Kaashoek, M. F., Karger, D., Karger, D., Morris, R., and Stoica, I. Looking up data in p2p systems. *Commun. ACM 46*, 2 (Feb. 2003), 43–48.

[3] Cardellini, V. Corso di ingegneria del web, università degli studi di roma. http://www.ce.uniroma2.it/courses/iw08, 2008. Accessed: 2019-05-07.

[4] GNU. GNUtella p2p file sharing. https://www.gnu.org/philosophy/gnutella.en.html, 2000. Accessed: 2019-05-07.

[5] Montresor, A. Materiale del corso di algoritmi e strutture dati. http://cricca.disi.unitn.it/montresor/teaching/asd/materiale/lucidi/, 2019. Accessed: 2019-05-07.