Homework

Ricci Flavia, Migliarini Matteo
Metodi probabilistici per l'informatica

June 12, 2022

1 Esercizio 1

A) Passo base (t=k): L'unico S_k possibile è N_k, quindi $\mathrm{P}(S_k = \mathrm{X}) = \mathrm{P}(S_k = N_k) = 1 = \binom{t}{k}^{-1} = \binom{k}{k}^{-1} = 1$. La proprietà è verificata.
Passo induttivo (t>k): Per ipotesi induttiva sappiamo che S_{t-1} è un sottoinsieme estratto u.a.r. da N_{t-1}. Si possono verificare due casi:

- l'elemento nuovo è salvato, ossia S_k e S_{k-1} differiscono per un elemento (prob. $= \frac{k}{t}$);
- l'elemento nuovo è scartato, ossia S_k e S_{k-1} sono uguali (prob. $= \frac{t-k}{t}$).

Sia X un generico insieme di k elementi tale che X $\subset \mathrm{N}_t$, e sia $\hat{\mathrm{X}}$ un insieme che differisce da X per uno specifico elemento. Nel primo caso si ha che:

$$\mathbb{P}(S_t = X \cap S_{t-1} = X) = \text{[teo. di Bayes] } \mathbb{P}(S_t = X | S_{t-1} = X) \cdot \mathbb{P}(S_{t-1} = X) =$$

$$= \frac{t-k}{t} \cdot \binom{t-1}{k}^{-1} = \binom{t}{k}^{-1}$$

Per l'altro caso invece osserviamo che:

$$\mathbb{P}(S_t = X | S_{t-1} = \hat{X}) = \text{[prob. di sostituire} \times \text{prob.di scartare uno specifico elemento tra k] } \frac{k}{t} \cdot \frac{1}{k} = \frac{1}{t}$$

e che il numero dei possibili $\hat{\mathrm{X}}$ è t-k. Infatti l'unico elemento variabile di $\hat{\mathrm{X}}$ (quello che sarà sostituito) può essere uno dei t-k elementi di $N_{k-1} \setminus \{$elementi fissi di $\hat{\mathrm{X}}\}$. Quindi:

$$\mathbb{P}(S_t = X \cap S_{t-1} = \hat{X}) = \mathbb{P}(S_t = X | S_{t-1} = \hat{X}) \cdot \mathbb{P}(S_{t-1} = \hat{X}) = \frac{1}{t} \cdot (t-k) \cdot \binom{t-1}{k}^{-1} = \binom{t}{k}^{-1}$$

Si ha $\mathbb{P}(S_t = X \cap S_{t-1} = X) = \mathbb{P}(S_t = X \cap S_{t-1} = X) = \binom{t}{k}^{-1}$. Dunque il fatto di differire o meno da S_{t-1} (ossia di contenere o meno l'ultimo elemento) non influisce sulla probabilità di osservarlo per cui: $\mathbb{P}(S_t = X) = \mathbb{P}(S_t = X \cap S_{t-1} = X) = \mathbb{P}(S_t = X \cap S_{t-1} = \hat{X}) = \binom{t}{k}^{-1}$
B) In generale sono dipendenti, ad esempio poichè S_t e S_{t+1} differiriscono al più per un elemento conoscendo S_t posso escludere di osservare alcuni S_{t+1}.

2 Esercizio 2

A) Il tempo di percorrenza deve almeno permettere a c pacchetti di attraversare lo stesso arco, cosa che non può avvenire in meno di c tempi, e deve inoltre permettere a un pacchetto di attraversare d archi, cosa che non può avvenire in meno di d tempi. Per cui esso è almeno $max(c,d) \geq \frac{1}{2}(c+d) = \Omega(c+d)$.
B) Assumiamo ragionevolmente che un pacchetto passi al più una volta sullo stesso arco. Sia $X_{e,t}$ il numero di pacchetti su e al tempo t. Sia dunque $k = \lceil \frac{\alpha c}{\log(Nd)} \rceil$ e $\mathbb{P}(\text{p passa su e al tempo t}) \leq \frac{1}{k}$ (l'evento può verificarsi solo se il pacchetto parte a un ben preciso istante).
Definiamo $Y_{e,t,p} \sim Bern(\frac{1}{k})$, allora $\sum_{p\in P}^{c} Y_{e,t,p} = X_{e,t} \sim Binomial(c, \frac{1}{k})$

$$\mathbb{P}(X_{e,t} > O(\log(Nd))) =$$

$$\mathbb{P}(\text{almeno } O(\log(Nd)) \text{ pacchetti passano su } e \text{ al tempo t}) \times (\text{combinazioni di } O(\log(Nd)) \text{ pacchetti tra c})$$

$$\leq \binom{c}{O(\log(Nd))}(\frac{O(\log(Nd))}{\alpha c})^{O\log(Nd)} \leq [\text{dis. di Stirling}](\frac{ce}{O(\log(Nd))})^{O(\log(Nd))}(\frac{O(\log(Nd))}{\alpha c})^{O\log(Nd)}$$

$$= (\frac{e}{\alpha})^{O(\log(Nd))}$$

C) Ogni pacchetto percorre al più d archi, quindi il numero totale degli attraversamenti è al più Nd. Utilizzando union bound sulla formula del punto b, si ottiene: $\mathbb{P}(anyX > O(\log(Nd))) \leq (\frac{e}{\alpha})^{O(\log(Nd))} \cdot Nd$ che vogliamo sia $< (\frac{1}{Nd})$. La disuguaglianza è soddisfatta se $\alpha > e\cdot(Nd)^{\frac{2}{O(\log(Nd))}} = e\cdot 2^{\frac{2\cdot\log(Nd)}{O(\log(Nd))}} = e\cdot 2^{O(1)} = O(1)$.
D) Sia P l'insieme dei pacchetti, $path_i$ e $start_i$ rispettivamente il percorso e il nodo d'inizio del pacchetto i-esimo.

Algorithm 1 Scheduling(P: piano orario originale)

```
N ← |P|
d ← max{l : ∀p ∈ P, l = |path_p|}
c ← max{c_e : c_e = |e ∈ path_p|, ∀p ∈ P}
for p ∈ P do
    delay_p ← random_between(1, ⌈50·c / log Nd⌉)
    prefix_p ← list[start_p,...,start_p] per delay_p volte
    path_p ← path_p ma preceduto da prefix_p
end for
return il nuovo P
```

Come dimostrato in (c) la proabilità che più di $O(\log(Nd))$ vogliano attraversare un certo arco in un certo momento t è al più $\frac{1}{Nd}$, ergo anche la probabilità che si creino code più lunghe di tale cifra è bassa in N. Nel caso peggiore ogni pacchetto dovrà percorrere al massimo una distanza d più il ritardo massimo $\lceil \frac{\alpha\cdot c}{\log Nd} \rceil$, e visto che ora bisogna rispettare il vincolo sugli archi, ogni pacchetto potrebbe dover aspettare al massimo per $O(\log Nd)$ per ogni suo step.

$$O(\log(Nd)\cdot(\lceil \frac{\alpha\cdot c}{\log Nd} \rceil + d)) = O(\alpha\cdot c + d\cdot\log(Nd)) = O(c + d\log(Nd))$$

3 Esercizio 3

A)
Ad ogni passo la lista si riduce di almeno un elemento (il pivot) e dunque l'algoritmo termina. Dimostriamo che le riduzioni applicate sono corrette:

Caso i = k-1: Assumiamo per assurdo che il pivot non sia il k-esimo elemento più grande. In tal caso il k-esimo elemento dovrebbe trovarsi in S_1 o in S_2, ma S_1 per costruzione contiene i primi k-1 elementi, quindi non può contenere il k-esimo e sempre per costruzione S_2 contiene solo elementi più grandi di tutti i k-1 elementi di S_1 e del pivot, ossia più grandi di almeno altri k elementi e dunque non può contenere il k-esimo.
Caso i $\geq$ k: Per costruzione S_1 contiene i primi i elementi e poiché i $\geq$ k tra questi primi i elementi ci sta anche il k-esimo, che (sempre per costruzione) coincide con il k-esimo elemento di S_1.
Caso i<k-1: Per costruzione S_1 non contiene il k-esimo elemento. Il pivot, essendo l'elemento immediatamente più grande degli elementi di S_1, è al più il (k-1)-esimo. Quindi il k-esimo elemento non può che stare in S_2. In particolare poichè S_2 è l'insieme di partenza da cui sono stati eliminati i primi i+1 elementi, il k-esimo elemento dell'insieme di partenza coincide con il (k-i-1)-esimo di S_2.
B) Il caso peggiore è quello con più ricorsioni e più confronti a ricorsione, ossia quello in cui il pivot è sempre un estremo. In tal caso $N = n + (n-1) + (n-2) + ... + 1 =$ [somm. di Gauss] $\frac{n(n+1)}{2} = O(n^2)$.
C) Definiamo una chiamata di successo come una chiamata che ha il pivot nel terzo mediano di S (di successo perché ci garantisce che la dimensione della lista su cui ricorrere sia contenuta entro un certo valore). Si ha che P(Successo) $= \frac{1}{3}$.
Sia n := |S iniziale|, dopo i successi $|S| \leq n(\frac{2}{3})^i$. Poiché l'algoritmo termina quando $|S| \leq 1$, termina quando $i \leq \log_{\frac{3}{2}}$n. Sia T_i = # iterazioni tra l'i-esimo e l'(i+1)-esimo successo. Si ha che $T_i \sim$ Geom(1/3), per cui E[T_i] = 3.

$$N = \sum_{i=0}^{\text{successi per terminare}} \text{confronti per successo} \Rightarrow \text{[linearità del valore atteso]}$$

$$\mathbb{E}[N] = \sum_{i=0}^{\text{successi per terminare}} E[\text{confronti per successo}] \leq \sum_{i=0}^{\text{successi per terminare}} (|\text{S all'i-esimo successo}|)\mathbb{E}[T_i] =$$

$$\leq \sum_{i=0}^{\log_{\frac{3}{2}} n} n(\frac{2}{3})^i 3 \leq 3n \sum_{i=0}^{\infty} (\frac{2}{3})^i = \text{[serie geometrica] } 3n\frac{1}{1-\frac{2}{3}} < 9n = O(n)$$

D) Per la disuguaglianza di Markov: $\mathbb{P}(N \geq a \cdot O(n)) \leq \frac{\mathbb{E}[N]}{a \cdot O(n)} = O(\frac{n}{a \cdot n}) \leq O(\frac{1}{a})$
Per ogni a di ordine di grandezza maggiore di $O(1)$, quando $n \to \infty$ la probabilità converge a 0. Perciò ad esempio se ci chiediamo quale sia la probabilità che l'algoritmo impieghi un tempo quadratico, allora $a = O(n)$, $\mathbb{P}(N \geq O(n^2)) \leq O(\frac{1}{n})$ che per $n \to \infty$ converge a 0.

4 Esercizio 4

A)

Algorithm 2 ClusteringCoefficient(V, E)

$C_G \leftarrow 0$
$n \leftarrow |V|$
for $v \in V$ **do**
  $N_v \leftarrow N(v)$
  $C_v \leftarrow 0$
  $\#coppie \leftarrow \binom{|N_v|}{2}$
  for $\{u, w\} \in N_v \times N_v$ **do**
    if $\{u, w\} \in E$ **then**
      $C_v \leftarrow C_v + \frac{1}{\#\text{coppie}}$
    end if
  end for
  $C_G \leftarrow C_G + \frac{C_v}{n}$
end for
return C_G

Siano $n = |V|$ e $m = |E|$ allora il costo asintotico di questo algoritmo è $O(n(m + mn^2)) = O(n^3 m)$ usando una rappresentazione con liste e $O(n^3)$ usando una matrice di adiacenza. Il caso peggiore è quando tutti i nodi hanno un arco verso tutti gli altri nodi (tutti i nodi sono vicini tra loro e $C_G = 1$).

B)

Per come è organizzato l'algoritmo, X_i è una variabile aleatoria di tipo Bernoulliano, $X_i \sim Bern(p)$. Infatti con una certa probabilità essa assume valore 1, o altrimenti 0. Ma quale è questa probabilità?

$$p = \mathbb{P}(\{u, w\} \in E | u \in N(v) \wedge w \in N(v))$$

Perché appunto ad X_i viene assegnato il valore 1 se u e w sono connessi da un arco, sapendo che u e w sono stati estratti uniformemente a caso da N(v). Perciò per il teorema di Bayes sulla probabilità condizionata:

$$\mathbb{P}(\{u, w\} \in E | u \in N(v) \wedge w \in N(v)) = \frac{\mathbb{P}(\{u, w\} \in E \wedge u \in N(v) \wedge w \in N(v))}{\mathbb{P}(u \in N(v) \wedge w \in N(v))} =$$

$$= \frac{\frac{|\{\{u,w\} \in E \wedge u \in N(v) \wedge w \in N(v)\}|}{\text{tutte le possibili coppie di nodi}}}{\frac{\text{tutte le possibili coppie di nodi tra i vicini di v}}{\text{tutte le possibili coppie di nodi}}} = \frac{|\{\{u, w\} \in E \wedge u \in N(v) \wedge w \in N(v)\}|}{\text{tutte le possibili coppie di nodi tra i vicini di } v} =$$

$$= \frac{|\{\{u, w\} \in E \wedge u \in N(v) \wedge w \in N(v)\}|}{\binom{|N(v)|}{2}} = C_v$$

Ora, sappiamo che $X_i \sim Bern(p)$ con $p \sim Uniform(C_v)$. Allora:

$$X = \frac{1}{l} \sum_{i=0}^{l} X_i = \frac{1}{l} \sum_{i=0}^{l} Bern(Uniform(C_v)) = Bern(Uniform(C_v))$$

$\mathbb{E}[X] = \mathbb{E}[Bern(Uniform(C_v))] = \mathbb{E}[Uniform(C_v)] = E[X_i] =$ [val. att. dell'uniforme] media di$(C_v) := C_G$

C)

$$\mathbb{P}((1-\epsilon)C_G \leq X \leq (1+\epsilon)C_G) = \mathbb{P}(-\epsilon C_G \leq X - C_G \leq \epsilon C_G) = \mathbb{P}(|X - C_G| \leq \epsilon C_G) = \mathbb{P}(|\sum_{i=0}^{l} X_i - lC_G| \leq \epsilon l C_G) \geq 1 - \delta$$

E ora basta applicare il metodo Montecarlo, sapendo che $\mathbb{E}[X] = C_G$ e $W_l = lX = \sum_{i=0}^{l} X_i$

$$\mathbb{P}(|W_l - lC_G| \geq l\epsilon C_G) = \mathbb{P}(|W_l - \mathbb{E}[W_l]| \geq \epsilon \mathbb{E}[W_l]) = \mathbb{P}(W_l - \mathbb{E}[W_l] \geq \epsilon \mathbb{E}[W_l]) + \mathbb{P}(W_l - \mathbb{E}[W_l] \leq \epsilon \mathbb{E}[W_l]) =$$

$$\leq [\text{dis. di Chernoff}] e^{-\frac{1}{3}\epsilon^2 \mathbb{E}[W_l]} + e^{-\frac{1}{2}\epsilon^2 \mathbb{E}[W_l]} < 2e^{-\frac{1}{3}\epsilon^2 \mathbb{E}[W_l]} = 2e^{-\frac{1}{3}\epsilon^2 l C_G} \leq \delta \Rightarrow l \geq \frac{3}{\epsilon^2 C_G} ln \frac{2}{\delta}$$