Algorithm Theory, Tutorial 6

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

January 2019

Ex1, Contention Resolution

Consider the contention resolution problem explained in the lecture with n processes and a single shared resource. We would like to calculate the expected number of time slots until every process has been successful at least once. For all integers $i \leq n$, let random variable T_i denote the smallest integer such that exactly i different processes are successful to access the resource in the first T_i time slots.

(a) Let t be an arbitrary time slot in $[T_j + 1, T_{j+1}]$ for an arbitrary integer $j < n$. What is the probability that some process becomes successful for the first time in time slot t?

Hint: The probability that some process is successful in a given time slot is $(1 - 1/n)^{n-1}$. *We have seen that this probability is approximately* $1/e$. *For simplicity, you can assume that this probability is exactly* $1/e$.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.
- Otherwise we wouldn't be in $[T_j + 1, T_{j+1}]$.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.
- Otherwise we wouldn't be in $[T_j + 1, T_{j+1}]$.
- The probability that any of the n processes proceeds in this time step is $\frac{1}{e}$.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.
- Otherwise we wouldn't be in $[T_j + 1, T_{j+1}]$.
- The probability that any of the n processes proceeds in this time step is $\frac{1}{e}$.
- We look for the probability that one process suceeds and that it is a process that has not suceeded before.

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.
- Otherwise we wouldn't be in $[T_j + 1, T_{j+1}]$.
- The probability that any of the n processes proceeds in this time step is $\frac{1}{e}$.
- We look for the probability that one process suceeds and that it is a process that has not suceeded before.
- If one process suceeds, the probability that this is one of the $n - j$ ones we are interested in is $\frac{n-j}{n}$ (at most one process can suceed at a time and all processes have the same probability).

Solution 1a

- We know that we are in $[T_j + 1, T_{j+1}]$
- This means that j processes have already suceeded at least once.
- We also know that the other $n - j$ processes have not suceeded before.
- Otherwise we wouldn't be in $[T_j + 1, T_{j+1}]$.
- The probability that any of the n processes proceeds in this time step is $\frac{1}{e}$.
- We look for the probability that one process suceeds and that it is a process that has not suceeded before.
- If one process suceeds, the probability that this is one of the $n - j$ ones we are interested in is $\frac{n-j}{n}$ (at most one process can suceed at a time and all processes have the same probability).
- The probability we are looking for thus is $\frac{n-j}{n} \cdot \frac{1}{e}$

Common pitfalls

- We know that we are in $[T_j + 1, T_{j+1}]$

Common pitfalls

- We know that we are in $[T_j + 1, T_{j+1}]$
- The probability that there still are $n - j$ processes that have not suceeded yet/ that no process has suceeded between T_j and now is 1.

Common pitfalls

- We know that we are in $[T_j + 1, T_{j+1}]$
- The probability that there still are $n - j$ processes that have not suceeded yet/ that no process has suceeded between T_j and now is 1.
- The exercise sheet's hint confused some and any.

1b

For all $i \leq n$, let random variable X_i be the number of rounds needed for the i^{th} process to succeed after exactly $i-1$ distinct processes have succeeded, i.e., $X_i := T_i - T_{i-1}$. Then, for an arbitrary integer $j \leq n$, what is the expected value of X_j?

1b, Solution

- Similar setting as in a), let

1b, Solution

- Similar setting as in a), let
- $P(X_i = 1) = \frac{n-i-1}{ne}$ (result from 1a)

1b, Solution

- Similar setting as in a), let
- $P(X_i = 1) = \frac{n-i-1}{ne}$ (result from 1a)
- $P(X_i = 2) = \left(1 - \frac{n-i-1}{ne}\right) \cdot \frac{n-i-1}{ne}$ (1 failure, then one success).

1b, Solution

- Similar setting as in a), let
- $P(X_i = 1) = \frac{n-i-1}{ne}$ (result from 1a)
- $P(X_i = 2) = \left(1 - \frac{n-i-1}{ne}\right) \cdot \frac{n-i-1}{ne}$ (1 failure, then one success).
- $P(X_i = k) = \left(1 - \frac{n-i-1}{ne}\right)^{k-1} \cdot \frac{n-i-1}{ne}$ ($k-1$ failures, then one success).

1b, Solution

- Similar setting as in a), let
- $P(X_i = 1) = \frac{n-i-1}{ne}$ (result from 1a)
- $P(X_i = 2) = \left(1 - \frac{n-i-1}{ne}\right) \cdot \frac{n-i-1}{ne}$ (1 failure, then one success).
- $P(X_i = k) = \left(1 - \frac{n-i-1}{ne}\right)^{k-1} \cdot \frac{n-i-1}{ne}$ ($k-1$ failures, then one success).
- X_i has geometric distribution with parameter $\frac{n-i-1}{ne}$.

1b, Solution

- Similar setting as in a), let
- $P(X_i = 1) = \frac{n-i-1}{ne}$ (result from 1a)
- $P(X_i = 2) = \left(1 - \frac{n-i-1}{ne}\right) \cdot \frac{n-i-1}{ne}$ (1 failure, then one success).
- $P(X_i = k) = \left(1 - \frac{n-i-1}{ne}\right)^{k-1} \cdot \frac{n-i-1}{ne}$ ($k-1$ failures, then one success).
- X_i has geometric distribution with parameter $\frac{n-i-1}{ne}$.
- The expected value of a geometric distribution is $param^{-1}$ thus

$$E[X_i] = \frac{ne}{n-i-1}$$

1c

What is the expected value of T_n, the time for all processes to succeed at least once?

1c, Solution

Due to the definition of T_i and X_i, considering $T_0 = X_0 = 0$, it holds that $T_n = \sum_{i=1}^{n} X_i$. Therefore, considering the linearity of expectation, it holds that

$$\begin{aligned}
E[T_n] &= E[X_1 + X_2 + \cdots + X_n] \\
&= E[X_1] + E[X_2] = \ldots E[X_n] \\
&= \sum_{i=1}^{n} \frac{en}{n-i+1} \\
&= \sum_{j=1}^{n} \frac{en}{j} \\
&= en \cdot H(n) \\
&\approx en \cdot \ln n
\end{aligned}$$

Ex 3: Randomized Independent Set

Let $G = (V, E)$ be a graph with n vertices and m edges. An independent set in a graph G is a subset $S \subseteq V$ of the nodes such that no two nodes in S are connected by an edge. Let $d := \frac{1}{n}\sum_{v \in V} \deg(v) = \frac{2m}{n}$ be the average node degree and consider the following randomized algorithm to compute an independent set S.

(I) Start with an empty set S. Then independently add each node of V with probability $1/d$ to S (you can assume that $d \geq 1$).

(II) The subgraph induced by S might still contain some edges and we therefore need to remove at least one of the nodes of each of the remaining edges. For this, we use the following deterministic strategy: As long as S is not an independent set, pick an arbitrary node $u \in S$ which has a neighbor in S and remove u from S.

It is clear that the above algorithm computes an independent set S of G.

2a

(a) Find a (best possible) lower bound on the expected size of S at the end of the algorithm. Your lower bound should be expressed as a function of n and d.
Hint: First compute the expected numbers of nodes in S and edges in $G[S]$ after Step (I) of the algorithm.

- Compute the expected number of nodes in S after the first step.

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step.

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step.
- Each edge $e \in E$ exists in $G[S]$ if and only if both of its adjacent nodes joined S in the first step.

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step.
- Each edge $e \in E$ exists in $G[S]$ if and only if both of its adjacent nodes joined S in the first step.
- This happens with probability $1/d^2$.

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step.
- Each edge $e \in E$ exists in $G[S]$ if and only if both of its adjacent nodes joined S in the first step.
- This happens with probability $1/d^2$.
- Thus we obtain

$$E[Y] = \sum_{e \in E} \frac{1}{d^2} = \frac{m}{d^2}.$$

- Compute the expected number of nodes in S after the first step.
- Let X be the random variable which indicates the size of S.
- By linearity of expectation we obtain

$$E[X] = \sum_{v \in V} \frac{1}{d} = \frac{n}{d}.$$

- Let Y be the random variable which denotes the number of edges in $G[S]$ after the first step.
- Each edge $e \in E$ exists in $G[S]$ if and only if both of its adjacent nodes joined S in the first step.
- This happens with probability $1/d^2$.
- Thus we obtain

$$E[Y] = \sum_{e \in E} \frac{1}{d^2} = \frac{m}{d^2}.$$

- We use that $m = dn/2$ and obtain

$$E[Y] = dn/2d^2 = n/2d$$

- In Expectancy we have now $n/2d$ edges.

- In Expectancy we have now $n/2d$ edges.
- Those have to be removed.

- In Expectancy we have now $n/2d$ edges.
- Those have to be removed.
- In the worst case (lower bound) removing one node only removes one edge. (draw example)

- In Expectancy we have now $n/2d$ edges.
- Those have to be removed.
- In the worst case (lower bound) removing one node only removes one edge. (draw example)
- The best possible lower bound for s thus is $X - Y$

- In Expectancy we have now $n/2d$ edges.
- Those have to be removed.
- In the worst case (lower bound) removing one node only removes one edge. (draw example)
- The best possible lower bound for s thus is $X - Y$
- We calculate

$$E[X - Y] = E[X] - E[Y] = \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d}$$

2b

Assume that the above algorithm has running time $T(n)$ and that it computes an independent set of size at least $\frac{n}{5d}$ with probability at least $\frac{1}{2}$. Show how to compute an independent set of size at least $\frac{n}{5d}$ with probability $1 - \frac{1}{n}$. What is the running time of your algorithm?

2b, solution

- Strategy: run algorithm multiple times and remember best found solution

2b, solution

- Strategy: run algorithm multiple times and remember best found solution
- If we run k times, the probability that our solution is still too bad is $\left(\frac{1}{2}\right)^k$.

2b, solution

- Strategy: run algorithm multiple times and remember best found solution
- If we run k times, the probability that our solution is still too bad is $\left(\frac{1}{2}\right)^k$.
- Thus the probability that we find a good enough solution is $1 - \left(\frac{1}{2}\right)^k$

2b, solution

- Strategy: run algorithm multiple times and remember best found solution
- If we run k times, the probability that our solution is still too bad is $\left(\frac{1}{2}\right)^k$.
- Thus the probability that we find a good enough solution is $1 - \left(\frac{1}{2}\right)^k$
- We solve

$$\left(\frac{1}{2}\right)^k \leq \frac{1}{n}$$

2b, runtime

- We have to run $k = \lceil \log_2(n) \rceil$ iterations of our algorithm

2b, runtime

- We have to run $k = \lceil \log_2(n) \rceil$ iterations of our algorithm
- This gives us a runtime of

$$k \cdot T(n) = \lceil \log_2(n) \rceil \cdot T(n)$$

Ex 3, 3-Coloring

The maximum 3-coloring problem asks for assigning one of the colors $\{1, 2, 3\}$ to each node $v \in V$ of a graph $G = (V, E)$ such that the number of edges $\{u, v\} \in E$ for which u and v get different colors is maximized. A simple randomized algorithm for the problem would be to (independently) assign a uniform random color to each node.
What is the expected approximation ratio of this algorithm?

Hint: Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$
- Thus the expected number of valid edges is $m \cdot \frac{2}{3}$

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$
- Thus the expected number of valid edges is $m \cdot \frac{2}{3}$
- Hint says Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$
- Thus the expected number of valid edges is $m \cdot \frac{2}{3}$
- Hint says Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.
- This ratio gets minimal when the optimal solution gets maximal.

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$
- Thus the expected number of valid edges is $m \cdot \frac{2}{3}$
- Hint says Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.
- This ratio gets minimal when the optimal solution gets maximal.
- Optimal solution is at most m (all edges are valid). For these graphs the expected Approximation Ratio is then $\frac{2}{3}$.

Solution Ex 3

- A given edge is valid, if both adjacent nodes have different colors. The probability for this is $\frac{2}{e}$
- Thus the expected number of valid edges is $m \cdot \frac{2}{3}$
- Hint says Consider the approximation ratio to be the minimum ratio of the algorithm solution to the optimal solution over all input instances.
- This ratio gets minimal when the optimal solution gets maximal.
- Optimal solution is at most m (all edges are valid). For these graphs the expected Approximation Ratio is then $\frac{2}{3}$.
- If you think, that this task is easy but its formulation is a bit cryptic, I share this thought.

Ex 4

Let $G = (V, E)$ with $n = |V|, m = |E|$ be an undirected, unweighted graph. Consider the following randomized algorithm: Every node $v \in V$ joins the set S with probability $\frac{1}{2}$. The output is $(S, V \setminus S)$.

a) What is the probability to actually obtain a cut?

Solution 4a

- For a cut we need to have at least one node in S and at least one in $V \setminus S$

Solution 4a

- For a cut we need to have at least one node in S and at least one in $V \setminus S$
- The probability that all nodes are in S is $\left(\frac{1}{2}\right)^n$

Solution 4a

- For a cut we need to have at least one node in S and at least one in $V \backslash S$
- The probability that all nodes are in S is $\left(\frac{1}{2}\right)^n$
- The probability that all nodes are in $S \setminus V$ is also $\left(\frac{1}{2}\right)^n$

Solution 4a

- For a cut we need to have at least one node in S and at least one in $V \setminus S$
- The probability that all nodes are in S is $\left(\frac{1}{2}\right)^n$
- The probability that all nodes are in $S \setminus V$ is also $\left(\frac{1}{2}\right)^n$
- Thus we get a cut with probability

$$1 - 2 \cdot \left(\frac{1}{2}\right)^n = 1 - \left(\frac{1}{2}\right)^{n-1}$$

4b + Solution

- For $e \in E$ let random variable $X_e = 1$ if e crosses the cut, and $X_e = 0$, else. Let $X = \sum_{e \in E} X_e$. Compute the expectation $E[X]$ of X.

4b + Solution

- For $e \in E$ let random variable $X_e = 1$ if e crosses the cut, and $X_e = 0$, else. Let $X = \sum_{e \in E} X_e$. Compute the expectation $E[X]$ of X.
- An edge crosses the cut iff both of its end points are on different sides of the cut. Thus

$$\forall e \in E : P(X_e = 1) = \frac{1}{2} = E[X_e]$$

4b + Solution

- For $e \in E$ let random variable $X_e = 1$ if e crosses the cut, and $X_e = 0$, else. Let $X = \sum_{e \in E} X_e$. Compute the expectation $E[X]$ of X.
- An edge crosses the cut iff both of its end points are on different sides of the cut. Thus

$$\forall e \in E : P(X_e = 1) = \frac{1}{2} = E[X_e]$$

- With the linearity of expectation we then directly get

$$E[X] = \sum_{e \in E} E[X_e] = m \cdot \frac{1}{2}$$

4c

Show that with probability at least $1/3$ this algorithm outputs a cut which is a $\frac{1}{4}$-approximation to a maximum cut (i.e. a cut of maximum possible size is at most 4 times as large).

Remark: For a non-negative random variable X, the Markov inequality states that for all $t > 0$ we have $P(X \geq t) \leq \frac{E[X]}{t}$.

Hint: Apply the Markov inequality to the number of edges **not** *crossing the cut.*

Solution 4c

- X is the number of edges in the cut

Solution 4c

- X is the number of edges in the cut
- Let $Y := m - X$ be the number of edges that are NOT in the cut (hint). $E[Y] = m - E[X] = \frac{m}{2}$

Solution 4c

- X is the number of edges in the cut
- Let $Y := m - X$ be the number of edges that are NOT in the cut (hint). $E[Y] = m - E[X] = \frac{m}{2}$
- Let $\mathcal{E}$ be the event that the algorithm produces a cut of size less than $\frac{m}{4}$. Then $\Pr(\mathcal{E}) = \Pr(X < \frac{m}{4}) = \Pr(Y \geq \frac{3m}{4})$.

Solution 4c

- X is the number of edges in the cut
- Let $Y := m - X$ be the number of edges that are NOT in the cut (hint). $E[Y] = m - E[X] = \frac{m}{2}$
- Let $\mathcal{E}$ be the event that the algorithm produces a cut of size less than $\frac{m}{4}$. Then $\Pr(\mathcal{E}) = \Pr(X < \frac{m}{4}) = \Pr(Y \geq \frac{3m}{4})$.
- Use the Markov inequality to calculate

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr(Y \geq \frac{3m}{4}) \\ &\leq \frac{E[Y]}{(3m/4)} \\ &= \frac{m/2}{3m/4} = \frac{2}{3}\end{aligned}$$

Solution 4c, Continuation

- The probability that the algorithm produces a cut of size less or equal $\frac{m}{4}$ is at most $\frac{2}{3}$.

Solution 4c, Continuation

- The probability that the algorithm produces a cut of size less or equal $\frac{m}{4}$ is at most $\frac{2}{3}$.
- $\Rightarrow$ With probability at least $1-\frac{2}{3}=\frac{1}{3}$ we get a cut of size more than $\frac{m}{4}$.

Solution 4c, Continuation

- The probability that the algorithm produces a cut of size less or equal $\frac{m}{4}$ is at most $\frac{2}{3}$.
- $\Rightarrow$ With probability at least $1-\frac{2}{3}=\frac{1}{3}$ we get a cut of size more than $\frac{m}{4}$.
- Obviously the number of edges that cross any cut is at most m. (Upper bound for optimal solution)

Solution 4c, Continuation

- The probability that the algorithm produces a cut of size less or equal $\frac{m}{4}$ is at most $\frac{2}{3}$.
- $\Rightarrow$ With probability at least $1-\frac{2}{3}=\frac{1}{3}$ we get a cut of size more than $\frac{m}{4}$.
- Obviously the number of edges that cross any cut is at most m. (Upper bound for optimal solution)
- $\Rightarrow$ Our algorithm outputs a $\frac{1}{4}$-approximation with probability $\frac{1}{3}$.

4d

- Show how to use the above algorithm to obtain a $\frac{1}{4}$-approximation of a maximum cut with probability at least $1-\left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$.
  Remark: If you did not succeed in (c) you can use the result as a black box for (d).

4d

- Show how to use the above algorithm to obtain a $\frac{1}{4}$-approximation of a maximum cut with probability at least $1-\left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$.
  Remark: If you did not succeed in (c) you can use the result as a black box for (d).
- Same idea as in exercise two: Run algorithm k times and remember best solution (biggest cut).

4d

- Show how to use the above algorithm to obtain a $\frac{1}{4}$-approximation of a maximum cut with probability at least $1-\left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$.
  Remark: If you did not succeed in (c) you can use the result as a black box for (d).
- Same idea as in exercise two: Run algorithm k times and remember best solution (biggest cut).
- Then the probability that we don't get $\frac{m}{4}$ edges or more is at most $(2/3)^k$, since all the repetitions are independent and the probability of failure of each repetition is at most $2/3$.

4d

- Show how to use the above algorithm to obtain a $\frac{1}{4}$-approximation of a maximum cut with probability at least $1-\left(\frac{2}{3}\right)^k$ for $k \in \mathbb{N}$.
  Remark: If you did not succeed in (c) you can use the result as a black box for (d).
- Same idea as in exercise two: Run algorithm k times and remember best solution (biggest cut).
- Then the probability that we don't get $\frac{m}{4}$ edges or more is at most $(2/3)^k$, since all the repetitions are independent and the probability of failure of each repetition is at most $2/3$.
- In other words, the probability that we get at least $\frac{m}{4}$ edges is *at least* $1-\left(\frac{2}{3}\right)^k$.