Algo sand 2

Kristian Degn Abrahamsen, 030701

November 2023

Contents

1 Problem 1

1.1 Question a (V)

Prove by induction on p that if A works correctly, then RANDPERM returns a random permutation of any given set S of $n = 2^p$ elements
Answer: first we define the base case. First, we can see that when $p = 0$ then the algorithm works, as we only have one element, i.e $n = 1$, and we can permute it in $n! = 1$ ways. then the probability of the permutation is

$$\frac{1}{n!} = \frac{1}{1}$$

Next, when $n = 2$, which corresponds to having 2^1 elements, having the set of elements $\{x_1, x_2\}$ the probability of getting any one of the sequences of elements is

$$\frac{1}{n!} = \frac{1}{2!} = \frac{1}{2}$$

since the only two options we have for the arrangements is

$$\{x_1, x_2\}, \{x_2, x_1\}$$

Either of these can serve as our basis. Now for the inductive step, we assume that this holds for $k = n \cdot 2$, or in other words $k = 2^{p+1}$. To prove this, we have to prove the following holds:
For $n \cdot 2$ the chance of getting any permutation is

$$\frac{1}{(n \cdot 2)!} = \frac{1}{k!} = \frac{1}{2^{p+1}!}$$

Now, when we have k elements, there are

$$\binom{k}{\frac{k}{2}}$$

ways to choose which elements are to be put in the left split and the right split. This is because, as A is generating a random subset of size 2^{k-1}, each of the k elements can be in the left or right split. And for each of the splits, there are per the induction hypothesis

$$(\frac{k}{2})!$$

Produced permutations on the split. This means that, for each permutation on the left split, there are $(\frac{k}{2})!$ permutations of the right split. We could of course also argue the other way around. This means that in total, for each of the $\binom{k}{\frac{k}{2}}$ choices of splits, there are produced $((\frac{k}{2})!)^2$ permutations. Putting it into together in terms, we have the total number of permutations for the k elements:

$$\binom{k}{\frac{k}{2}} \cdot ((\frac{k}{2})!)^2$$

To simplify this, we have

$$\binom{k}{\frac{k}{2}} = \frac{k!}{(k-\frac{k}{2})! \cdot \frac{k}{2}!} = \frac{k!}{(\frac{k}{2}!)^2}$$

Inserting this into the expressions for calculating how many permutations are producing

$$\frac{k!}{(\frac{k}{2}!)^2} \cdot (\frac{k}{2}!)^2 = k! = (n \cdot 2)! = 2^{p+1}!\Big]$$ correct but why are they equally likely?

In conclusion, there are produced $(2n)!$ permutations for the $2n$ elements, meaning that RANDPERM returns a random permutation of any given set S of $n = 2^p$ elements. I.e, per induction, this has been proved for all powers of 2.

1.2 Question b ✓

Describe an implementation of the algorithm A and prove that this will return a random subset of size $\frac{S}{2}$ of S when the input is a set S with an even number of elements.

Answer: To do this, we can for instance choose between the two algorithms we have seen in class:

1. randomize in place
2. randomize by sorting

I will just go with the **randomize by sorting**, where we in this case just return the first half of the array.

The algorithm looks like

```
permuteBySorting(A)
    n <- |A|
    for i in 1 to n do
        p(i) <- random(1,n^3)
    sort A using p as keys
    return first half of A
```

That is, assign a certain priority to each member of A and sort on the priorities. If there are any clashes, i.e two elements gets the same random priority, then we will simply redo the algorithm. The probability that two elements gets the same priority is $\frac{1}{n^3}$, due to the fact that we fix the first element, and then the 2nd element has to hit the exact same priority, when it can be assigned to n^3 priorities. It can easily be proven that the probability of a clash is $< \frac{1}{n}$, which means that the probability of having no clashes is $\geq 1 - \frac{1}{n} = \frac{n-1}{n}$ which means the expected number of times we have to run the algorithm to have no clashes, is $\frac{1}{p} = \frac{1}{\frac{n-1}{n}} = \frac{n}{n-1}$ since this can be described by a random indicator variable, following a geometric distribution. I.e the expected number of times we have to run the algorithm is $\lceil \frac{n}{n-1} \rceil = 2$. Now lets prove that this algorithm will actually return a random subset, of half of the elements. Let's start by looking at the

then do it ☺

entire A. For this, we have to prove that any permutation out of the $n!$ possible, the chance of getting any permutation is $\frac{1}{n!}$. It is fairly easy to see that the algorithm will in fact return a random permutation, since the first element of A can be in n spots, and the spot is chosen at random. The 2nd element can be in $n-1$ spots, as it cannot collide with the first element, else the algorithm will start over, and so on.
If we want to prove it more formally, we are looking at the probability that each element is at the exact position of where it was before the sorting. that is

$$p(X_1 \cap X_2 \cap X_3 \cap \ldots \cap X_n) = p(X_1)p(X_2|X_1)P(X_3|X_1 \cap X_2)\ldots p(X_n|X_1 \cap X_{n-1})$$

The probability of the event X_1 is then

$$p(X_1) = \frac{1}{n}$$

As out of the n elements, this element has to be mapped to the smallest priority.

$$P(X_2|X_1) = \frac{1}{n-1}$$

As out of now $n-1$ elements, this element has to be mapped to the smallest priority. And so it continues. We end up with the following statement

$$p(X_1 \cap X_2 \cap X_3 \cap \ldots \cap X_n) = \frac{1}{n} \cdot \frac{1}{n-1} \cdot \frac{1}{n-2} \cdot \ldots \frac{1}{1} = \frac{1}{n!}$$

Meaning that the probability of getting any permutation is $\frac{1}{n!}$ which proves, that when looking at the entirety of A a random permutation is generated. If we now look at the half of A, each of the n elements has an equal likelihood of ending up at the first position of the array. When the first position is chosen each of the $n-1$ elements can be in the 2nd spot and so on. This continues until the size of the set is half of the original set, meaning that it is random which of the n elements are in the first half of the set, therefore a random subset is returned.

2 Problem 2 ✓

Solve the recurrence equation $a_n = a_{n-1} + 2a_{n-2}$ with initial conditions $a_0 = 4$ and $a_1 = 6$
Answer: We start with rewriting the equation to the form

$$r^n = r^{n-1} + 2r^{n-2}$$

We will then divide on both sides by r^{n-2}

$$r^2 = r + 2$$

Next, we form the characteristic equation, by subtracting both sides by $r + 2$

$$r^2 - r - 2 = 0$$

When solving for r we get the roots:

$$r_1 = 2,\ r_2 = -1$$

Note, that these are two distinct roots, so I will follow the calculation for this. Other equations are used when dealing with double roots.
As by theorem 1 in the lecture notes, we can find the solution to a_n by solving

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

Then we get the equations

$$a_0 = \alpha_1 r_1^0 + \alpha_2 r^0 = \alpha_1 + \alpha_2 = 4$$

$$a_1 = \alpha_1 r_1 + \alpha_2 r_2 = 6$$

This is two equations with two unknowns, and we can start by isolating α_1 in the equation for a_0 and we get:

$$\alpha_1 = 4 - \alpha_2$$

We substitute α_1 by the this in the equation for a_1 and get the equation:

$$(4 - \alpha_2)r_1 + \alpha_2 r_2 = 4r_1 - \alpha_2 r_1 + \alpha_2 r_2 = 6$$

We can then insert the known values for the roots and start solving for α_2

$$4 \cdot 2 - \alpha_2 \cdot 2 + \alpha_2 \cdot (-1) = 8 - 2\alpha_2 - \alpha_2 = 8 - 3\alpha_2 = 6$$

$$\alpha_2 = \frac{6-8}{-3}$$

We can insert this into the equation for α_1

$$\alpha_1 = 4 - \alpha_2 = 4 - \frac{6-8}{-3} = \frac{10}{3}$$

Now we have all values for the equation for a_n, giving us the final formula for a_n:

$$a_n = \frac{10}{3}2^n + \underset{=-\frac{2}{3}}{\frac{6-8}{-3}}(-1)^n$$

3 Problem 3

3.1 Question a ✓

Suppose you have i of the coupons already for some even number $0 \leq i < n$. What is the probability of having $i + 2$ coupons after the next step?
Answer: at first we define the entire space, which is

$$\binom{n}{2}$$

Since we can choose two coupons from the n possible coupons in this number of ways, and do note that we don't care about the ordering, therefore the use of the *choose* formula. Next, there are

$$\binom{n-i}{2}$$

ways to pick the last coupons, when we have already collected i of them, and we have to choose 2 in the same round. In conclusion, the probability of having $i + 2$ coupons after the next step is

$$\frac{\binom{n-i}{2}}{\binom{n}{2}}$$

3.2 Question b: ✓

Prove that when you have exactly $i < n$ different coupons, for some even integer i, the expected number of times you have to collect two coupons before you will have $i + 2$ coupons is $\frac{\binom{n}{2}}{\binom{n-i}{2}}$
Answer: this follows the geometric distribution, Rosen section 7.4.5, since we may try an infinite amount of times before moving to the next step, meaning the expected number of times we have to collect 2 distinct coupons in this round is

$$\frac{1}{p} = \frac{1}{\frac{\binom{n-i}{2}}{\binom{n}{2}}} = \frac{\binom{n}{2}}{\binom{n-i}{2}}$$

3.3 Question c ✓

Show that the expected number of coupons you need to collect before you have all coupons is $O(n^2)$ you may use that $\frac{1}{q(q-1)} = \frac{1}{q-1} - \frac{1}{q}$
Answer: as calculated above, we expect to be in the $i'th$ round for

$$\frac{\binom{n}{2}}{\binom{n-i}{2}}$$

rounds. But in each round we will collect 2 coupons, so the expected number of coupons we have to collect in each round is

$$2\frac{\binom{n}{2}}{\binom{n-i}{2}}$$

We can sum up all of the coupons collected in each phase i by

$$\sum_{i=0}^{\frac{n}{2}-2} 2\frac{\binom{n}{2}}{\binom{n-2i}{2}}$$

As $2\binom{n}{2}$ is not dependent on i, i.e it is merely a constant, we can take it out of the sum.

$$2\binom{n}{2}\sum_{i=0}^{\frac{n}{2}-2}\frac{1}{\binom{n-2i}{2}}$$

$$= 2\binom{n}{2}\sum_{i=0}^{\frac{n}{2}-2}\frac{1}{\frac{1}{2}(n-2i-i)(n-2i)}$$

$$= 2\binom{n}{2}\sum_{i=0}^{\frac{n}{2}-2}\frac{1}{(n-2i-i)(n-2i)}\cdot 2$$

$$= 4\binom{n}{2}\sum_{i=0}^{\frac{n}{2}-2}\frac{1}{(n-2i-i)(n-2i)}$$

Now, the terms in the summation will be:

$$\frac{1}{n-1}-\frac{1}{n}+\frac{1}{n-3}-\frac{1}{n-2}+\frac{1}{n-5}-\frac{1}{n-4}\cdots\frac{1}{n-n+4-1}-\frac{1}{n-n+4}$$

Which simplifies to the alternating harmonic series

$$\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n} = ln\ 2$$

Therefore

$$4\binom{n}{2}\sum_{i=0}^{\frac{n}{2}-2}\frac{1}{(n-2i-i)(n-2i)} = 4\frac{1}{2}n(n-1)\cdot ln\ 2$$

$$= (2n^2-2n)\cdot ln\ 2$$

Where n^2 is the dominating expression meaning the amount of coupons we expected to collect is

$$O(n^2)$$

4 Problem 4

4.1 Question a ✓

What is the expected number R of clauses we will satisfy if there are m clauses? You must show how to obtain your answer.
Answer: The probability that a clause is not satisfied is $(\frac{1}{2})^k = \frac{1}{2^k}$ since there are 2^k ways to give truth assignment to a clause with k variables. This is true since we can use the product rule, since the truth assignment to the 2nd does not depend on the 1st, which is true for all of the k variables. There is only 1 of these assignments which will make the clause false, which is the assignment to the k variables, where all variables are assigned the value *false*. Now, the probability of a clause being satisfied is $1 - \frac{1}{2^k} = \frac{2^k-1}{2^k}$. This is because the probability of the opposite case, when there are only two outcomes, is $pr(E) = 1 - pr(!E)$.
We now define a random indicator variable X_i being 1 when the clause i is satisfied, and 0 otherwise. As there are m clauses, we will have m random indicator variables, where we can use linearity expectation which will look like this:

$$E(X) = E(\sum_{i=0}^{m} X_i) = \sum_{i=0}^{m} E(X_i) = \sum_{i=0}^{m} \frac{2^k-1}{2^k} = m \cdot \frac{2^k-1}{2^k}$$

This means that the expected number of clauses satisfied, R, is

$$R = m \cdot \frac{2^k-1}{2^k}$$

4.2 Question b ✓

Prove that there exists a truth assignment that makes at least R of the clauses true.
Answer: as stated by Kleinberg and Tardos, page 725 below theorem 13.14, Since we use a random variable "there must be at some point at which it assumes some value at least as large as its expectation". This is, of course, true, since if the random variable never met its expectation in any of the cases, then the expectation would be smaller.

4.3 Question c

Describe a Las Vegas algorithm A which given a K-sat formula F will find a truth assignment that makes at least R of the clauses in F true.
Answer: since we have proved there exists a truth assignment which makes R of the clauses true, a Las Vegas algorithm could be implemented in such a way, that it simply tries at random truth assignment of the k variables until the assignment satisfies R clauses, and then returns it. The same idea is used in Kleinberg and Tardos page 726 **Further analasys: Waiting to Find a Good Assignment**, so I will give it a reference. ✓

4.4 Question d ✓

Give an upper bound on the expected running time of A. Hint: follow the idea in Kleinberg and Tardós section 13.4.
Answer: we have shown that the probability of satisfying at least R clauses is

$$m \cdot \frac{2^k - 1}{2^k}$$

If we now can show that this is at least p the expected number of trials we have to perform of the algorithm above, is $\frac{1}{p}$. Now, denote a value $i = 0, 1, 2, \ldots, m$ as being the number of clauses satisfied by the algorithm. This means we can calculate the expected number of clauses satisfied by

$$\sum_{i=0}^{m} ip_i = m \cdot \frac{2^k - 1}{2^k}$$

We are interested in the cases where $i \geq m \cdot \frac{2^k-1}{2^k}$ since we want to satisfy at least R of the clauses, as stated in *Question b*. We can now write up the expected value in the sum format provided above

$$m \cdot \frac{2^k - 1}{2^k} = \sum_{i < m \cdot \frac{2^k-1}{2^k}} ip_i + \sum_{i \geq m \cdot \frac{2^k-1}{2^k}} ip_i$$

Now let m' be the largest number strictly less than $m \cdot \frac{2^k-1}{2^k}$. If we insert this in the above equation, in the left sum this will only increase the equations value. We will further insert m instead of i on the right sum, which will further only increase the value outputted by the equation. We now have

$$m \cdot \frac{2^k - 1}{2^k} \leq \sum_{i < m \cdot \frac{2^k-1}{2^k}} m'p_i + \sum_{i \geq m \cdot \frac{2^k-1}{2^k}} mp_i$$

Now observe that $\sum_{i < m \cdot \frac{2^k-1}{2^k}} p_i = 1 - p$ since the probability of satisfying at least $m \cdot \frac{2^k-1}{2^k}$ of the clauses is p, so the probability of satisfying less than this is the above. Vice versa, we can also see that

$$\sum_{i \geq m \cdot \frac{2^k-1}{2^k}} p_i = p$$

due to what has been stated.

$$m \cdot \frac{2^k - 1}{2^k} \leq \sum_{i < m \cdot \frac{2^k-1}{2^k}} m'p_i + \sum_{i \geq m \cdot \frac{2^k-1}{2^k}} mp_i = m'(1-p) + mp \leq m' + mp$$

Now observe that $mp \geq m \cdot \frac{2^k-1}{2^k} - m'$ and $m \cdot \frac{2^k-1}{2^k} - m' \geq \frac{1}{2^k}$ since m' is strictly smaller than $\frac{2^k-1}{2^k}$ times some other natural number. therefore

$$p \geq \frac{m \cdot \frac{2^k-1}{2^k} - m'}{m} \geq \frac{1}{m2^k}$$

Now that we have a lower bound on p, and we know that the expected number of trials performed by the algorithm is $\frac{1}{p} = \frac{1}{\frac{1}{m2^k}} = m2^k$ meaning the number of trials before the algorithm finds an assignment which satisfies at least $m \cdot \frac{2^k-1}{2^k}$ of the clauses, is it at most $m2^k$, or in other terms $O(m2^k)$

5 Problem 5

5.1 Question a ✓

prove that the expected number of balls in a fixed box B_i is m
Answer: The probability that a ball thrown hits in box B_i is $\frac{1}{m}$ since the ball will hit a random box when thrown. We make a random indicator variable, X_{ij} taking the value 1 if the j'th-ball lands in B_i and 0 otherwise. This means that

$$E(X_{ij}) = p = \frac{1}{m}$$

We can now use linearity of expectation on the m^2 different balls being thrown, as each throw is independent of the other throws.

$$E(X) = E(\sum_{j=0}^{m^2} X_{ij}) = \sum_{j=0}^{m^2} E(X_{ij}) = m^2 \frac{1}{m} = m$$

5.2 Question b (✓)

Use Chebyshev's inequality to bound the probability that the number of balls in box B_i is more than 50% away from its expected value
Answer: here we want to provide the bound on the deviation of more than $0.5m$, which is because the expected value is m. This means we put $r = 0.5m$ in the provided formula for chebyshevs inequality in Rosen, theorem 8 page 517. and we obtain the statement:

$$p(\mid X(s) - E(X) \mid \geq 0.5m) \leq \frac{V(X)}{(0.5m)^2}$$

We now want to calculate the variance of the random variable, which is calculated by

$$V(X) = E(X^2) - E(X)^2$$

We can already calculate $E(X)^2$ as the expected value is m:

$$E(X)^2 = m^2$$

calculating $E(X^2)$ will be

$$E(X^2) = E((X_{i0} + X_{i1} + E_{i2} \dots E_{im^2})^2)$$

we can rewrite this to

$$E(X^2) = E(\sum_{j=0}^{m^2} X_{ij} + 2 \cdot \sum_{0 \leq j < k \leq m^2} X_{ij} X_{ik})$$

Using linearity of expectation we get

$$E(X^2) = \sum_{j=0}^{m^2} E(X_{ij}) + 2 \cdot \sum_{0 \leq j < k < m^2} E(X_{ij} X_{ik})$$

correct here ↙

$$= m + m^2(m^2-1)(\frac{1}{m})^2$$

$$= m + \frac{m^4 - m^2}{m^2}$$

$$= m^2 + m - 1 \quad ✓$$

The value of $E(X_{ij}X_{ik})$ is $\frac{1}{m} \cdot \frac{1}{m} = (\frac{1}{m})^2$ since the two throws j and k are independent, and the two random indicator variables takes the value 1 with the probability p where $p = \frac{1}{m}$.

Summing up, and inserting into the formula we get the bound of the probability that the number of balls in box B_i is more than 50% away from its expected value m is

$$p(\mid X(s) - m \mid \geq 0.5m) \leq \frac{m^2 + m - 1 - m^2}{(0.5m)^2} = \frac{m-1}{0.25m^2} \quad ✓$$

5.3 Question c ✓

Use the Chernoff bounds (13.42) and (13.43) from Kleinberg-Tardós to bound the probability that the number of balls in box B_i is more than 50% away from its expected value.

Anser: to provide the probability that the random variable X is 50% larger than its expected value, we use *13.42*:

$$Pr[X > (1+\delta)\mu] < \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{\mu}$$

Inserting $\delta = 0.5$ and $\mu = m$

$$Pr[X > 1.5m] < \left[\frac{e^{0.5}}{1.5^{1.5}}\right]^{m} = 0.89745^m$$

This is the probability that X takes values 50% larger than its expected value. Now we also want to bound the probability that X takes values 50% smaller than its expected value. We do this by following *theorem 13.43*.

$$Pr\left[X < (1-\delta)\mu\right] < e^{-\frac{1}{2}\mu\delta^2}$$

inserting values as before

$$Pr\left[X < (1-\delta)\mu\right] < e^{-\frac{1}{2}m0.5^2} = e^{-\frac{1}{2}\cdot\frac{1}{4}m} = e^{-\frac{m}{8}}$$

Now we combine both of the bounds to achieve the bound of X being more than 50% away from its expected value. We do this by adding the two events together, to get the entire event.

$$p[|X - m| > 0.5m] \leq 0.89745^m + e^{-\frac{m}{8}}$$

5.4 Question d (✓)

Compare the two bounds above and explain the difference. Which bounds is best as m gets large?
Answer:

m	Chernoff	Chebyshev
$m = 10$	0.625	0.36
$m = 100$	0.0000237	0.0396
$m = 1000$	$1.0239 \cdot 10^{-47}$	0.004

Table 1: Comparison of Chernof and Chebyshev Bounds for different values of m.

As we can see, when you compare the chernoff bound to the chebyshev bound, chernoff provides a much better bound. An example is the fact that Chernoff bound is already providing a better bound when $m = 100$ than what Chebyshev does at $m = 1000$. Therefore as m gets large Chernoff is better. However, it can also be seen that when m is small, Chebyshevs bound is providing a better bound than Chernoff.

Why does this happen?

5.5 Question e ✓

Use your bound from Question c and the union bound to show that when $m = 100$ the probability that there is any box which has less than 50 or more than 150 balls is less than 1%.
Answer: the probability that one box has less than 50 balls or more than 150 balls, i.e $m = 100$ is

$$0.0000237\%$$

As seen in table 1. We want to apply the union bound on all of the boxes, meaning we can calculate this by

$$p(b_1) + p(b_2) + p(b_3) \dots p(b_m) = 100 \cdot 0.0000237 = 0.00237$$

This means that the probability that any box, when $m = 100$, has less than or more than 150 balls is less than 1%.

6 Problem 6

6.1 Question a ✓

Give a short description of the Edmonds-karp algorithm for finding a maximum flow and illustrate the algorithm on the network in Figure 1. Remember to say how much you augment by along each path. **In order to make correction easier you should do this exactly as follows:** each new augmenting path should not only be a shortest path in the current residual network, it should

also have the smallest name lexicographically. That is, the path $s14t$ is the first augmenting path.
You should draw the residual network each time when there are no more augmenting paths of the current shortest path length. Thus after listing a set of augmenting paths of length 3 (according to the rule above) so that no more augmenting paths of length 3 can be found (in the current residual network), you give the correct residual network and then go on to the next set of paths (which will have length 4, as you will see).
Answer: The Edmonds-Karp algorithm is just like the Ford-Folkerson algorithm, but the paths in the residual network, is the **shortest (s, t)-path**. This leads to the polynomial running time $O(|V||E|^2)$ instead of the worst-case running time for Ford-Folkerson which is $O(|f||E|)$ where f is the maximum flow. As seen in the examples provided in the lectures, this can be much worse than Edmonds-Karp, when there is an arc with only 1 as its capacity, or really just any small capacity compared to the maximum flow, as its capacity, and this arc is in the augmenting path for each iteration. Edmonds-Karp takes advantage of the breadth-first-search algorithm, for finding the shortest path in the residual network.
The original network is $N = (V, E, c)$ and the residual network is $N_f = (V, E_f, c_f)$
The residual network revolves around the calculation:

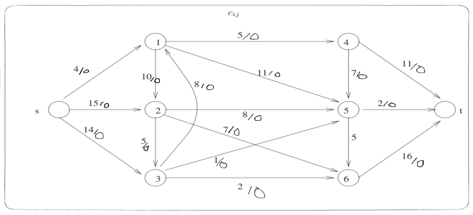
- if $(u, v) \in E$ $c_f(u, v) = c(u, v) - f(u, v)$
- if $(v, u) \in E$ $c_f(u, v) = f(v, u)$
- otherwise $c_f(u, v) = 0$

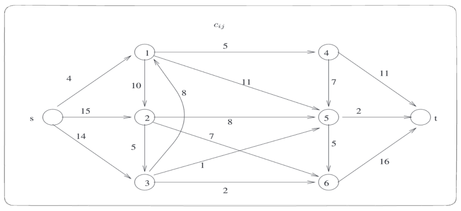
and $E_f = \{(u, v) \in VxV | c_f(u, v) > 0\}$
The most left network is the network, and the network on the right is the corresponding residual network

1. The residual network for the network looks exactly like the network in first iteration
2. The shortest lexicographical path is $s14t$, and the smallest capacity of all the arcs is 4
3. The shortest lexicographical path is $s25t$, and the smallest capacity of all the arcs is 2
4. the shortest lexicographical path is $s26t$, and the smallest capacity of all the arcs is 7
5. the shortest lexicographic path is $s36t$, and the smallest capacity of all the arcs is 2
6. the shortest lexicographic path is $s256$ and the smallest capacity of all the arcs is 5

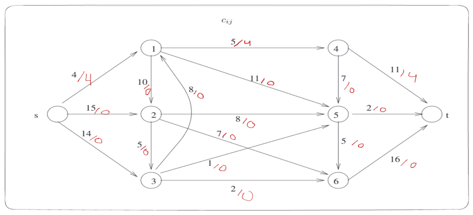
7. the shortest lexicographic path is $s314t$ and the smallest capacity of all the arcs is 1

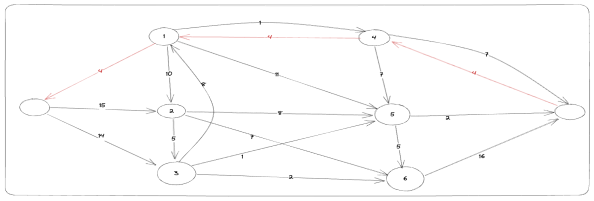
1


c_{ij}
s
t
4/0
5/0
10/0
8/0
11/0
7/0
11/0
15/0
8/0
2/0
14/0
7/0
5/0
1/0
5
16/0
2/0

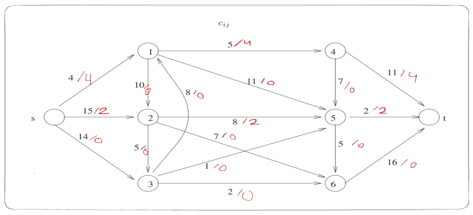

c_{ij}
s
t
4
5
10
8
11
7
11
15
8
2
14
7
5
1
5
16
2

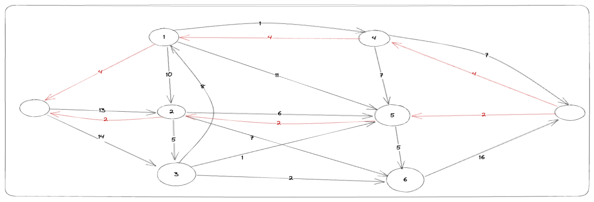
2


c_{ij}
s
t
4/4
5/4
10/0
8/0
11/0
7/0
11/4
15/0
8/0
2/0
14/0
7/0
5/0
5/0
1/0
16/0
2/0

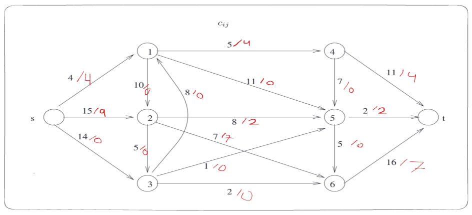


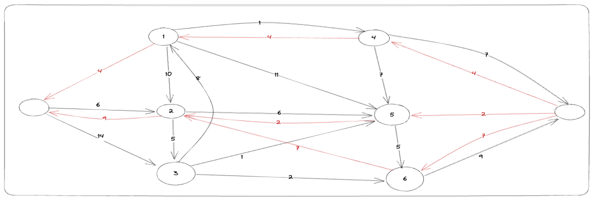
3


c_{ij}
s
t
4/4
5/4
10/0
8/0
11/0
7/0
11/4
15/2
8/2
2/2
14/0
7/0
5/0
5/0
1/0
16/0
2/0

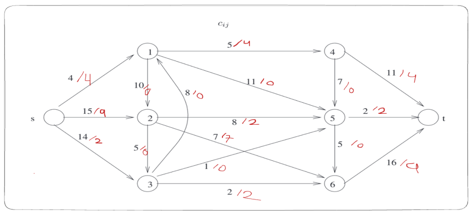


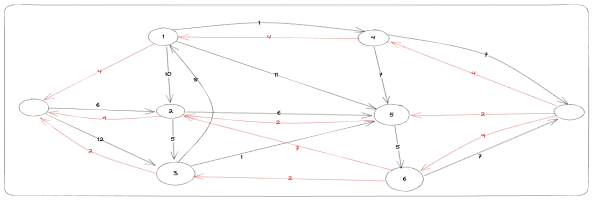
4


c_{ij}
s
t
4/4
5/4
10/0
8/0
11/0
7/0
11/4
15/9
8/2
2/2
14/0
7/7
5/0
5/0
1/0
16/7
2/0

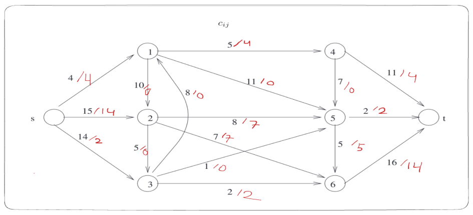


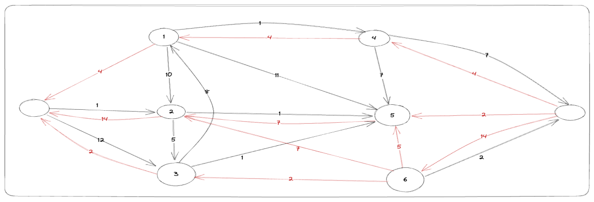
5


c_{ij}
s
t
4/4
5/4
10/0
8/0
11/0
7/0
11/4
15/9
8/2
2/2
14/2
7/7
5/0
5/0
1/0
16/9
2/2

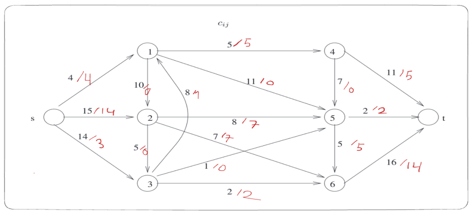


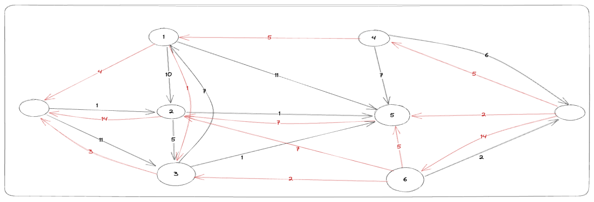
6


c_{ij}
s
t
4/4
5/4
10/0
8/0
11/0
7/0
11/4
15/14
8/7
2/2
14/2
7/7
5/0
5/5
1/0
16/14
2/2



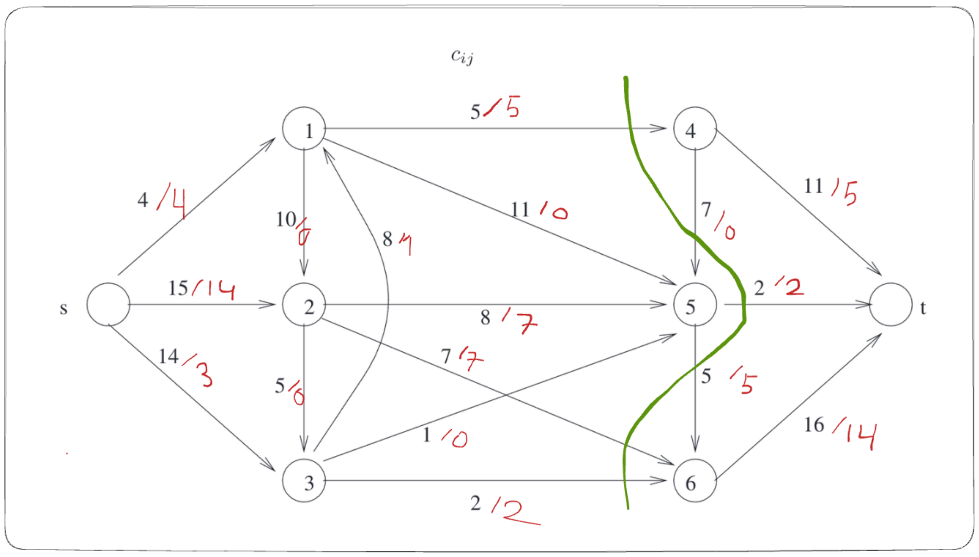
7


c_{ij}
s
t
4/4
5/5
10/0
8/4
11/0
7/0
11/5
15/14
8/7
2/2
14/3
7/7
5/0
5/5
1/0
16/14
2/2



6.2 Question b ✓

Give the values on every arc of the resulting maximum flow f^*, give its value, and also a minimum cut whose capacity shows that f^* is a maximum flow.
Answer:



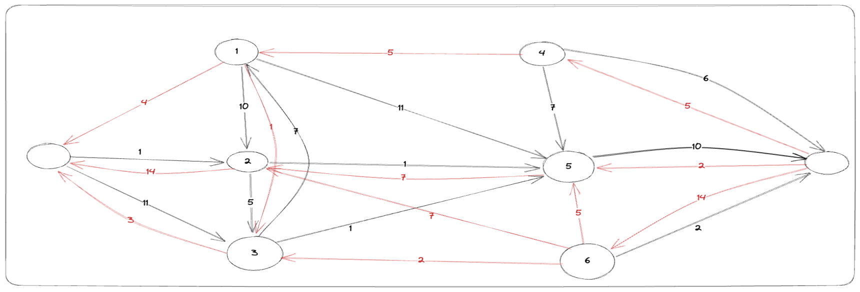
The maximum flow is the balance of s, and negative balance of t. This means that, since there is coming $4 + 14 + 3 = 21$ flow out of s, then the maximum flow is 21. This can be seen as we find a minimum cut in the network, which matches the flow, proving that this is a maximum flow since the cut value is $5 + 2 + 5 + 7 + 2 = 21$

6.3 Question c ✓

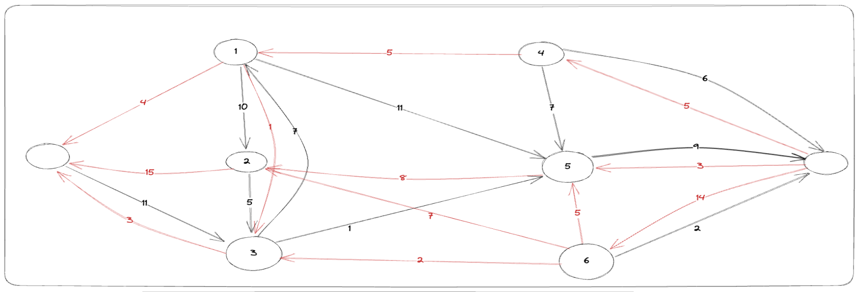
Suppose now that we increase the capacity of the arc from vertex 5 to t to 12. Use your final residual network above to say what the value of a new maximum flow will be and give a new cut that shows that this value is indeed the maximum.
Answer: The flow is again the balance of s, or negative the balance of t. $4 + 15 + 11 = 30$

1. shortest path is $s25t$ and smallest capacity of all the arcs is 1
2. shortest path is $s35t$ and smallest capacity of all the arcs is 1
3. shortest path is $s315t$ and smallest capacity of all the arcs is 7
4. final residual network

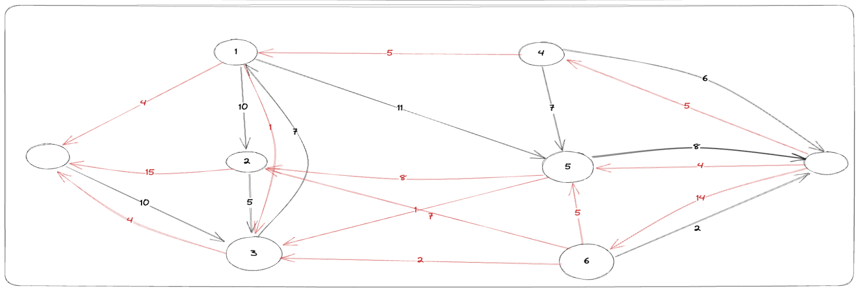
1



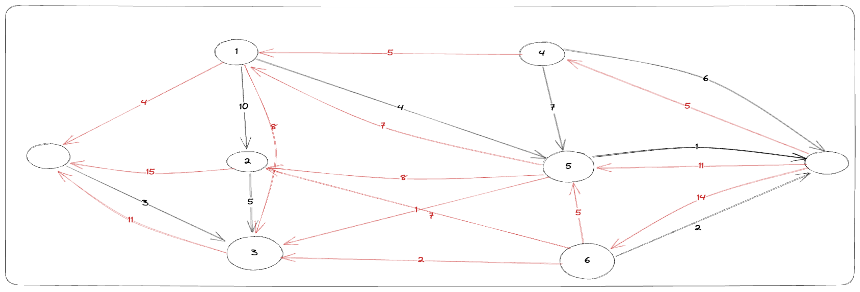
2

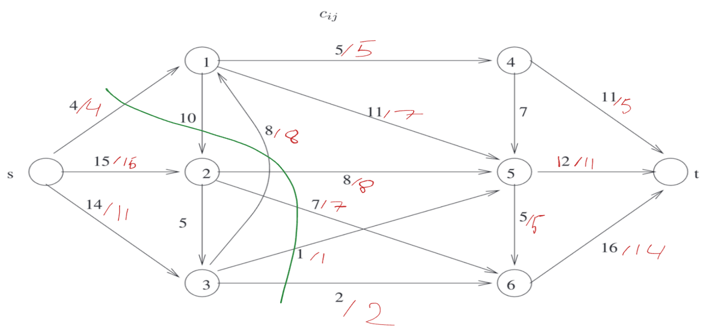


3



4





Well done!