APC Special Assignment 2

Nicolas Muntwyler - 17-915-737 - Gruppe C

November 2019

Exercise 1

a) We first note that $i, j \in \{1, ..., n\}$. Therefore $\epsilon_{i,j} \in \{\epsilon_{1,1}, ..., \epsilon_{n,n}\}$
With that we can rewrite:

$$E[\underbrace{\epsilon_{i,\pi_1(i)}\epsilon_{i,\pi_2(i)}\cdots\epsilon_{i,\pi_{2k+1}(i)}}_{:=A}] = E[(\epsilon_{1,1})^{k_{1,1}}(\epsilon_{1,2})^{k_{1,2}}\cdots(\epsilon_{n,n})^{k_{n,n}}]$$

with:

$$k_{i,j} = \text{number of times } \epsilon_{i,j} \text{ appeared in A} \qquad i, j \in \{0, 1, ..., n\}$$

Note that if for a specific $\epsilon_{i,j}$ we have that it never appears, we choose $k_{i,j} = 0$ which results in $\epsilon_{i,j}^{k_{i,j}} = \epsilon_{i,j}^0 = 1$. And since multiplying with 1 doesn't change A, we indeed have equality.

Now since we have 2k+1 permutations, we also have that the number of epsilons in A is 2k+1. Therefore we get:

$$S = \sum_{i,j\in\{1,...,n\}} k_{i,j} = 2k + 1$$

Again since for a not represented epsilon $\epsilon_{i,j}$ we chose $i, j = 0$ and adding 0 doesn't change the sum, we indeed have equality.

Now since 2k+1 is odd, we ask ourself how we can get an odd number. We know that if we add two numbers and the result is odd, me must have that one of the summands has to be odd. Additionally we have that 0 is even. If we combine these two facts we get that we must have at least one $k_{i,j}$ that is uneven and that the corresponding $\epsilon_{i,j}$ appears at least once in A (because $k_{i,j} \neq 0$). Let that epsilon be $\epsilon_{p,q}$ with corresponding $k_{p,q}$ (p,q $\in \{1, ..., n\}$)

Before we continue we note that two different epsilons are independent (They are u.a.r $\in \{-1, 1\}$). Hence $\epsilon_{1,1}$ is independent of $\epsilon_{1,2}$, ..., $\epsilon_{n,n}$. The same holds for all other epsilons. Therefore we have:

$$E[(\epsilon_{1,1})^{k_{1,1}}(\epsilon_{1,2})^{k_{1,2}}\cdots(\epsilon_{p,q})^{k_{p,q}}\cdots(\epsilon_{n,n})^{k_{n,n}}]$$

$$= E[(\epsilon_{1,1})^{k_{1,1}}] \cdot E[(\epsilon_{1,2})^{k_{1,2}}]\cdots E[(\epsilon_{p,q})^{k_{p,q}}]\cdots E[(\epsilon_{n,n})^{k_{n,n}}]$$

With claim1 we get:

$$E[(\epsilon_{1,1})^{k_{1,1}}] \cdot E[(\epsilon_{1,2})^{k_{1,2}}]\cdots E[(\epsilon_{p,q})^{k_{p,q}}]\cdots E[(\epsilon_{n,n})^{k_{n,n}}]$$

$$= E[(\epsilon_{1,1})^{k_{1,1}}] \cdot E[(\epsilon_{1,2})^{k_{1,2}}]\cdots 0 \cdots E[(\epsilon_{n,n})^{k_{n,n}}] = 0$$

And with that we showed:

$$E[\epsilon_{i,\pi_1(i)}\epsilon_{i,\pi_2(i)}\cdots\epsilon_{i,\pi_{2k+1}(i)}] = 0$$

Claim1: $E[(\epsilon_{p,q})^{k_{p,q}}] = 0$

Proof:
Since the epsilons can only be 1 or -1 we get:

$$E[(\epsilon_{p,q})^{k_{p,q}}] = 1^{k_{p,q}} \cdot Pr[(\epsilon_{p,q})^{k_{p,q}} = 1] + (-1)^{k_{p,q}} \cdot Pr[(\epsilon_{p,q})^{k_{p,q}} = -1]$$

We know that $k_{p,q}$ is odd. Hence $(-1)^{k_{p,q}} = (-1)$. Obviously $1^{k_{p,q}} = 1$.
And for the same reason we have $(\epsilon_{i,j})^{k_{i,j}} = \epsilon_{i,j}$, since $\epsilon_{i,j}$ is either 1 or -1.
With that we get:

$$E[(\epsilon_{p,q})^{k_{p,q}}] = 1 \cdot Pr[(\epsilon_{p,q})^{k_{p,q}} = 1] + (-1) \cdot Pr[(\epsilon_{p,q})^{k_{p,q}} = -1]$$

And since $\epsilon_{i,j}$ u.a.r $\in \{1, -1\}$ we get:

$$E[(\epsilon_{p,q})^{k_{p,q}}] = 1 \cdot \frac{1}{2} + (-1) \cdot \frac{1}{2} = 0$$

b)

$$E[(det(B)^{2k+1}]$$

$$\overset{(1)}{=} E[(\sum_{\pi \in S_n} (sign(\pi) \prod_{i=1}^{n} b_{i,\pi(i)}))^{2k+1}]$$

$$\overset{(2)}{=} E[(\sum_{\pi \in S_n} (sign(\pi) \prod_{i=1}^{n} a_{i,\pi(i)} \cdot \epsilon_{i,\pi(i)}))^{2k+1}]$$

$$\overset{(3)}{=} E[\sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in S_n} (sign(\pi_1) \prod_{i=1}^{n} a_{i,\pi_1(i)} \cdot \epsilon_{i,\pi_1(i)}) \cdots (sign(\pi_{2k+1}) \prod_{i=1}^{n} a_{i,\pi_{2k+1}(i)} \cdot \epsilon_{i,\pi_{2k+1}(i)}).]$$

$$\overset{(4)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in S_n} E[(sign(\pi_1) \prod_{i=1}^{n} a_{i,\pi_1(i)} \cdot \epsilon_{i,\pi_1(i)}) \cdots (sign(\pi_{2k+1}) \prod_{i=1}^{n} a_{i,\pi_{2k+1}(i)} \cdot \epsilon_{i,\pi_{2k+1}(i)})]$$

$$\overset{(5)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in Z} \left(sign(\pi_1) \underbrace{\prod_{i=1}^{n} a_{i,\pi_1(i)}}_{=1} \cdots sign(\pi_{2k+1}) \underbrace{\prod_{i=1}^{n} a_{i,\pi_{2k+1}(i)}}_{=1} \right) \cdot E[\prod_{i=1}^{n} \epsilon_{i,\pi_1(i)} \cdots \prod_{i=1}^{n} \epsilon_{i,\pi_{2k+1}(i)}]$$

$$\overset{(6)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in Z} (sign(\pi_1) \cdots sign(\pi_{2k+1})) \cdot E[\prod_{i=1}^{2k+1} \epsilon_{1,\pi_i(1)} \cdots \prod_{i=1}^{2k+1} \epsilon_{n,\pi_i(n)}]$$

$$\overset{(7)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in Z} (sign(\pi_1) \cdots sign(\pi_{2k+1})) \cdot E[\prod_{i=1}^{2k+1} \epsilon_{1,\pi_i(1)}] \cdots E[\prod_{i=1}^{2k+1} \epsilon_{n,\pi_i(n)}]$$

$$\overset{(8)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in Z} (sign(\pi_1) \cdots sign(\pi_{2k+1})) \cdot 0 \cdots 0$$

$$\overset{(9)}{=} \sum_{\pi_1,\pi_2,\ldots,\pi_{2k+1} \in Z} 0$$

$$\overset{(10)}{=} 0$$

(1) Leibniz formula for the determinant

(2) Definition of Matrix B

(3) Expanding the multiplication of 2k+1 sums. We are using: $(\sum_{i=1}^{n} a_i)(\sum_{j=1}^{n} b_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j$

(4) Linearity of Expectation

(5) Since the *sign*'s are numbers we can take them out of the expectation. We did this in our exercise session (group A, 27.november). Now let $Z := \{\pi \in S_n | a_{1,\pi(1)} a_{2,\pi(2)} \cdots n, \pi(n) = 1\}$ and then we only take the permutations with $\pi \in Z$. We therefore have that $\prod_{i=1}^{n} a_{i,\pi_1(i)} = 1$. We can do this, because if $\prod_{i=1}^{n} a_{i,\pi_1(i)} \neq 1$, we must have $\prod_{i=1}^{n} a_{i,\pi_1(i)} = 0$, and therefore it would just add 0 to the expected value. (So we just leave it out)

(6) Just reordering the order of multiplying the epsilons. (Commutative)

(7) Epsilons of different rows are independent

(8) Using part a) with i = 1, ..., i = n.

(9) Multiplying with 0 results to 0

(10) Sum of 0's is 0

Exercise 2

a) Before we begin we would like to make some definitions to ease the explanations:

- We can numerate the pairs (i,j) $\in$ P. So we just have one number l $\in \{1..|P|\}$. So the last pair would have number $|P|$.
- Let $m := |P|$
- To each pair we associate a flow. So we will speak of the l-th flow (corresponds to l-th pair in the numeration). Therefore we will speak of x_i when we talk about the i-th flow.
- We also numerate the arcs from 1 to n.
- When we speak of "having a better solution" we mean that we get a lower w (width of circle).

We define the integer linear program LP as:

minimize

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ 1-x_1 \\ 1-x_2 \\ \vdots \\ 1-x_m \\ w \end{pmatrix}$$

subject to

$$\left(\begin{array}{c|c|c} & & -1 \\ & & -1 \\ \mathbf{A} & \mathbf{B} & -1 \\ & & \vdots \\ & & -1 \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ 1-x_1 \\ 1-x_2 \\ \vdots \\ 1-x_m \\ w \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$x_j \in \{0,1\} \qquad 1 \leq j \leq m$$

$$A_{ij} := \begin{cases} 1 & \text{if arc i would be used, if the flow j goes counterclockwise. } (1 \le i \le n, 1 \le j \le m) \\ 0 & \text{otherwise} \end{cases}$$

$$B_{ij} := \begin{cases} 1 & \text{if arc i would be used, if the flow j goes clockwise. } (1 \le i \le n, 1 \le j \le m) \\ 0 & \text{otherwise} \end{cases}$$

Keep in mind:
Note that in a matrix constraint system of the form Ax≤b, we would need, that the vector x is just a vector of variables. Hence we can't have $(1 - x_i)$ in the x-vector. Therefore we would replace the $(1 - x_i)$ with y_i and add additional constraint that $y_i = (1 - x_i)$. If we would also want to bring this into a "≤" form we could replace it with: $y_i \le (1 - x_i)$ and $(1 - x_i) \le y_i$.
But since we would still give the same explanations/proofs as with the LP from above and the LP above is much more readable and intuitively, we will use the LP from above.

We now want to prove that this LP models the problem P. To prove this we show:

(i) "optimal solution to LP corresponds to an optimal solution in P"

(ii) "optimal solution to P corresponds to an optimal solution in LP"

Because: (i) & (ii) $\Longrightarrow$ "LP models P"

Proof of (i)
optimal solution to LP corresponds to an optimal solution in P $\Longleftrightarrow$
The assignment of the variables $x_1, ..., x_m, w$ (such that they maximize the LP) will also give an optimal solution in P.

Let A_{LP} be the assignment of the variables $(x_1, ..., x_m, w)$, such that it is an optimal solution for the LP. We now want to prove that for that assignment it is also an optimal solution for P.

Proof by Contradiction:
So lets assume we have a different assignment A_P (i.e. $A_{LP} \neq A_P$), which results in a better solution for P as with the A_{LP} assignment.
We now want to prove, that if we assign our variables in the LP with the A_P assignment, all constraints will still hold and hence be a better feasible solution (since the w is lower), which is a contradiction to the assumption that A_{LP} is the optimal assignment for the LP.
To prove that the A_P assignment fulfills all constraints we once again use a contradiction prove. Let's assume they don't. Then for at least one constraint the "≤" isn't true. In particular:

$$a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m) - w > 0$$

$$\Longleftrightarrow$$

$$a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m) > w$$

But after fact1: $a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m)$ corresponds to the width of arc i. And Therefore the width of the i-th arc would be greater than w. This would lead that A_P doesn't fulfill P, which is a contradiction.

Proof of (ii)
optimal solution to P corresponds to an optimal solution in LP $\Longleftrightarrow$
The assignment of the variables $x_1, ..., x_m, w$ such that they give an optimal solution in P, will also give an optimal solution in LP, hence maximize it.

Let A_P be the assignment of the variables $(x_1, ..., x_m, w)$, such that is is an optimal solution for P. We now want to prove that for that assignment it is also an optimal solution for the LP.

Proof by Contradiction:
So lets assume we have a different assignment A_{LP} (i.e. $A_{LP} \neq A_P$), which results in a better solution for LP as with the A_P assignment.
We now want to prove, that this A_{LP} assignment could also be used in P and produce a better solution in P as A_P, which is a contradiction to that A_P is optimal in P.
To prove that A_{LP} will produce a better solution in P we show that the arc in P with the maximum width is lower than in A_P. Let arc i be that arc.
We know that:

$$a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m) - w \leq 0$$

$$\Longleftrightarrow$$

$$a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m) \leq w$$

And form fact1 $a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m)$ corresponds to the width of arc i. Therefore arc i has a lower (or equal) width than w. Note that this w comes from the A_{LP} assignment, which produces per assumption a better solution than LP, hence has a lower w. This implies that arc i has a lower width than the width from A_P, and since arc i is the arc with maximal width, we would conclude that with that assignment we have a lower w, and hence a better solution.

fact1: $a_{i1}x_1 + ... + a_{im}x_m + b_{i1}(1 - x_1) + ... + b_{im}(1 - x_m)$ corresponds to the width of arc i

Proof:
We know that if x_i is 0 then $(1 - x_i)$ is 1. Which makes sense, since for every flow we either have it flow clockwise(0) or counterclockwise(1). So when we multiply $a_{ij}x_j$ we get a 1 only if for the flow j, the arc i gets used if we go counterclockwise and flow actually goes counterclockwise. Therefore we only add flow j to the width of arc i if and only if the j-th flow actually uses arc i and in counterclockwise order. We can make the same argument for the clockwise order with $b_{ij}(1 - x_j)$.
So put together we count flow j to the width of arc i iff flow j actually uses arc i.
Note that we never "doublecount" (counterclockwise and clockwise) because x_i or $(1 - x_i)$ has to be 0.

b) We now switch back in the Notation to x_{ij} for flow (i,j) instead of x_k (as in sub part a)).

WLOG we always assume that $j > i$. Let's first define our LP that we want to use:

maxmize
- $c^T x$

subject to:
- $x_{ij} \leq 0$
- $x_{ij} \leq n - 2(j - i)$

$c :=$ $(1_1, ..., 1_m)^T$ (remember that $|P| = m$)
$x :=$ vector of x_{ij} for all pairs $(i, j) \in P$

We want to prove:

(i) LP returns 0 for x_{ij} if and only if the flow from i to j is shorter (less involved arcs) if we go clockwise.

(ii) LP returns a negative number for x_{ij} if and only if the flow from i to j is strictly shorter if we go counterclockwise.

If (i) and (ii) hold we can construct an Algorithm that first runs the LP, and then for the assignment of the LP reassigns the x_{ij} as follows:

- If $x_{ij} = 0$, then $x_{ij} = 0$ ($\rightarrow$ clockwise)
- If $x_{ij} < 0$, then $x_{ij} = 1$ ($\rightarrow$ counterclockwise)

With that assignment we have that:

(I) If $x_{ij} = 0$, then the flow from i to j is shorter if we go clockwise

(II) If $x_{ij} = 1$, then the flow from i to j is shorter if we go counterclockwise

We then want to prove that from (I) and (II) follows that with that assignment (A_A) we have a maximal width of an arc less that 2 times the Optimal possible value.

Proof:
Let's assume we have an arc i that has a width $w_i > 2\text{OPT} = 2w$, if we choose the assignment from our algorithm, hence A_A. This means that 2w + k flows go over arc i with $k \in \{1, 2, , ..\}$ with A_A. In the optimal assignment A_{OPT} arc i has $\leq$ w flows that go over him. Therefore We get that ($\geq$) w+k flows had to be changed from A_{OPT} to A_A, such that arc i has at least w + (w + k) = 2w + k flows that go over him.
We now argue, that since we changed the direction of at least w + k flows from A_{OPT} to A_A, now at least w + k flows don't flow over arc x = $(mod_n(\lceil \frac{n}{2} \rceil + i))$ anymore. Geometrically arc x is the arc that is on the other side of the circle. Or in other words, the arc that is the farthest from arc i. (The rounding is for breaking ties in uneven cases). But if this is true (which we prove later), we have a contradiction. Because if now w+k flows don't go over arc x anymore, this would mean that in the optimal solution we actually had to have at least w+k flows that go over arc x. However we then would have that arc x would have a width bigger than the maximum width, which is a contradiction.
Therefore we must have, that arc i has a width $\leq$ 2OPT. Hence A_A returns an assignment which is at worst 2 times as bad as the optimal solution.

Lastly lets prove that the w+k flows that got redirected such that they go over arc i, had to flow over arc x in the optimal solution.
This comes from the fact that A_A is the assignment where the flow always chooses the shorter path (least amount of used arcs). So if clockwise only uses 8 arcs and counterclockwise 10, we choose that this flow goes clockwise.
We notice that if we would choose both direction (clockwise and counterclockwise) we would span the whole circle. And since the shorter path consists of $\leq \lfloor \frac{n}{2} \rfloor$ arcs, the longer path consists of $\geq \lceil \frac{n}{2} \rceil$ arc. Or in other words the shorter path has always $\leq$ half of all arcs, and the longer path always $\geq$ half of all arcs. With this we can conclude that for every arc y on the shorter path, the opposite arc of arc y, has to be in the longer path. (Because the distance between those two is always at least $\lfloor \frac{n}{2} \rfloor$).
Now since arc i from above is in the shorter path (due to the construction of our assignment), we get that arc x (the opposite arc) has to lie in the longer path. And with that we can conclude that if a flow now goes over i (and before it didn't), he must have gone over x before.

Proof of (i)
We need to show both directions:

1.) If the LP return 0 for x_{ij}, the flow from i to j is shorter if we go clockwise

2.) If the flow from i to j is shorter if we go clockwise, the LP returns 0 x_{ij}.

So lets prove this:

1.) If $x_{ij} = 0$ we have that:
$0 \leq n - 2(j - i) \iff j - i \leq n - (j - i) \iff$ clockwise path is shorter than counterclockwise

2.) clockwise path is shorter than counterclockwise $\iff j - i \leq n - (j - i) \iff 0 \leq n - 2(j - i)$ And since we maximize such that we want to have every single x_{ij} as big as possible, we choose the biggest number for x_{ij} that we can choose, which is 0.

Proof of (ii)
We need to show both directions:

1.) If the LP return $x_{ij} < 0$, the flow from i to j is strictly shorter if we go counterclockwise
2.) If the flow from i to j is strictly shorter if we go counterclockwise, the LP returns $x_{ij} < 0$.

So lets prove this:

1.) The second constraint gives: $x_{ij} \leq n - 2(j - i) \iff x_{ij} + (j - i) \leq n - (j - i)$. At this point we would like to assign x_{ij} a number as big as possbible since we maximize (hence we would like x_{ij} to be 0). But since we have that $x_{ij} < 0$ we obviously cant. This means that the following doesn't hold:
$0 + (j - i) \leq n - (j - i)$
And therefore the following must hold:
$(j - i) > n - (j - i)$.
Which means nothing more than, that it is shorter to go counterclockwise than clockwise.
2.) If the flow is shorter when we go counterclockwise we get that $n - (j - i) < (j - i)$. Which means that $0 < n - 2(j - i)$. And since x_{ij} needs to fulfill this constraint we must have, that $x_{ij} < 0$.

Note that we used: (j-i) $\iff$ Length of path in clockwise order. And therefore n-(j-i) is the length of the path in counterclockwise order. We can do this since we assumed that $j > i$ (wlog).

Exercise 3

Some given facts:

- All half-spaces are upper
- No boundary hyperplane is vertical
- The common intersection among more than d boundary hyperplanes is empty
- for $2 \leq j \leq d$ the intersection of any j boundary hyperplanes is (d-j)-dimensional. (If d = 5, then the intersection of 3 hyperplanes is 2 dimensional)
- Lowest vertex exists

a) We first prove that:
"The return value v from the algorithm Random-LP(n,d) is indeed the lowest vertex for the polyhedron $\bigcap_{0 \leq j \leq n} h_j^*$. (With n arbitrary)"

Proof (by induction over d):

Base (d=2): The if-statement on the first line is fulfilled since d=2. Therefore we just return Random-2D-LP, which is guaranteed to return the lowest vertex.
Induction Hypothesis: The value v (= return value of Random-LP(n,d)) is indeed the lowest vertex for the polyhedron $\bigcap_{0 \leq j \leq n} h_j^*$.
Induction step: We need to show that Random-LP(i,d+1) returns a lowest vertex of the polyhedron $\bigcap_{0 \leq j \leq n} h_j^*$ To show that we need to prove that after the for-loop, that v is a lowest vertex of the polyhedron $\bigcap_{0 \leq j \leq n} h_j^*$.
To do that, we us yet an other induction:
Let v_i be v at the end of the (i-d)th iteration (of the for-loop) (where $d - 1 \leq i \leq n$, because we start with $h_0, ..., h_{d-1}$). We then want to prove:
"v_i is a lowest vertex of the polyhedron $\bigcap_{0 \leq j \leq i} h_j^*$"

Proof (by induction on i):

Base (i=d-1) This is the assignment of v before we even start the loop. We assign v to be the intersection point among the d hyperplanes, $h_0, h-1, ..., h_{d-1}$. Since they are all upper, this vertex has to be the lowest. Otherwise, this means v is not in every h_i, and we could still move into the direction of the h_i that v isn't inside. And since the half-spaces are all upper, we would get to a lower point, which is a contradiction that we had a lowest point. This means that the lowest point must be in the intersection of all the h_i with $(0 \leq i \leq d-1)$. Now since v_i is in the intersection we know that v_i must be a lowest point.
Induction Hypothesis: v_i is a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i} h_j^*$
Induction Step (i → i+1): We know that the assignment of v (for now its equal to v_i) is a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i} h_j^*$. Now let's go through the iteration of the loop:
If the if-statement on line 8 fails (hence we skip to line 11), we trivially get that $v \in h_{i+1}^*$ and since $v \in \bigcap_{0\leq j\leq i} h_j^*$ (from inner I.H.) we get that $v \in \bigcap_{0\leq j\leq i+1} h_j^*$. And v would only get higher (upper?), we have that v (and since we are at the end of the loop: $v = v_{i+1}$) must be a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i+1} h_j^*$.
If the if-statement on line 8 succeeds (hence we run line 9 and 10), we then just run Random-LP((i+1)-1,d-1), where thanks to outer I.H. we get the lowest vertex returned of $\bigcap_{0\leq j\leq i+1} h_j^*$ (Note that we can do this because we can use the black-box, which guarantees that one lowest vertex in the primal space is till a lowest vertex in the dual space and vice-versa. Hence we don't have to care about the conversion). Now since we now are at the end of the loop-iteration we have that v_{i+1} is indeed a lowest vertex of $\bigcap_{0\leq j\leq i+1} h_j^*$.

With this inner induction proven, we can use it with i=n to get that v_n (which is v after the n-th iteration, hence just v) is a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq n} h_j^*$, which we needed to prove.

Now we finally can begin to prove the statement we were actually meant to prove in this exercise. We want to show:

"If v $\notin h_i^*$, then a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i} h_j^*$ lies in h_i."

Proof:

Since v is the return value of a Random-LP we know that v must be a lowest vertex of a certain polyhedron (H_i^*). But since $v \notin h_i^*$, we conclude that v is a lowest vertex of a polyhedron that doesn't include h_i^*. This means that the lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i} h_j^*$ can't be v. With that said we denote with v' the vertex that is a lowest vertex of the polyhedron $\bigcap_{0\leq j\leq i} h_j^*$.

Remember that in the 2D-case we imagined a line segment s connecting v and v'. We now want to do the same but in higher dimension. Therefore we have a line l (in higher dimension) which connects v and v'.

Since v $\notin h_i^*$ but $v' \in h_i^*$, l has to "cross in higher dimension" with h_i at some point. We call that intersection point p (i.e. the point that is on l and in h_i).

We know that v must be lower than v'. Because v was chosen in a previous iteration as a lowest vertex and since v' is in $\bigcap_{0\leq j\leq i} h_j^*$, v' would also be in $\bigcap_{0\leq j\leq k} h_j^*$, with $k \leq i$ and therefore we could have just chosen v'. But we didn't. We chose v, which means v must have been lower than v'.

So if we go "along" l from v to v' we only get higher and higher (or upper and upper?). Therefore we want the first feasible point (point that is also in h_i^*) that we can take along our journey from v to v' (since we dont want to go higher). And this happens as soon as we reach points that are also in h_i^*. The first point that fulfills that will be p.
Therefore we conclude that $v' = p$.

Since $v' = p$ and p is in h_i we conclude that v' lies in h_i.

b) We denote by $T(n,d)$ the running time of Random-LP with n half-spaces and d dimensions.
At first site our running time is the (n-d) recursive calls from the for loop, where the recursive calls recuse again them self, until we read d=2. With only this assumption we would get a running time that is bigger than $O(d!n)$.
However we notice that the if-statement on line 8 of the algorithm must have a success probability, which improves our running time. We now want to take a closer look fo that if-statement. Let:

Pr[round i is bad] $\iff$ We execute line 9 and 10
Pr[round i is good] $\iff$ We skip to line 11

On line 4 we generate a random permutation of the half-spaces and take one after the other into our H_i^*. So in some iterations v is still in the intersection even with the new half-space. To calculate the success probability of that happening we do a backwards-analysis:

If we are in iteration i, and v be the lowest point of the intersection of the 0,..,i half-spaces, hence (i+1) half-spaces. Now the probability, that v changes is at most $\frac{d}{i+1}$ since v is defined of at most d boundary hyperplanes (which is because the common intersection among more than d boundary hyperplanes is empty). So if we have i+1 half-spaces and we would delete one, v only changes if we delete a half-space which its corresponding boundary hyperplane is one of the hyperplanes that define v. And since we have at most d such defining hyperplanes we indeed have:

$$Pr[\text{v changes in i-th iteration}] \leq \frac{d}{i+1}$$

And with that we get that:

$$Pr[\text{round i is bad}] = Pr[\text{v changes in i-th iteration}] \leq \frac{d}{i+1}$$

Now the expected running time of the Random-LP can be written recursively as:

$$E[T(n,d)] = E\left[\sum_{i=d}^{n} Pr[\text{round i is bad}] \cdot T(i-1,d-1) + Pr[\text{round i is good}] \cdot O(1)\right]$$

The O(1) is because we don't have a recursive call if the if-statement is good (skips to line 11), hence no call for Random-2D-LP.
Since Pr[round i is good] ≤ 1 we get that $Pr[\text{round i is good}] \cdot O(1)$ is just a constant that we call c_1.
Together with the fact that Pr[round i is bad] $\leq \frac{d}{i+1}$ we get:

$$E[T(n,d)] \leq E\left[\sum_{i=d}^{n} \frac{d}{i+1} \cdot T(i-1,d-1) + c_1\right]$$

Linearity of expectation yields:

$$E[T(n,d)] \leq \sum_{i=d}^{n} \frac{d}{i+1} \cdot E[T(i-1,d-1)] + c_1$$

Using this derived formula we want to use induction (over d) to prove:

Proof: "$E[T(n,d)] = O(d!n)$"

Base case (d=2): $E[T(n,2)] = O(n) = O(2n) = O(2!n)$
Induction Hypothesis: $E[T(n,d-1)] = O((d-1)!n)$

Induction Step ($(i-1) \to i$):

$$\begin{aligned}
E[T(n,d)] &\leq \sum_{i=d}^{n} \frac{d}{i+1} \cdot E[T(i-1,d-1)] + c_1 \\
&\overset{I.H.}{\leq} \sum_{i=d}^{n} \frac{d}{i+1} \cdot c_2(d-1)!(i-1) + c_1 \\
&= \sum_{i=d}^{n} \frac{i-1}{i+1} \cdot c_2 d! + c_1 \\
&= c_2 d! \sum_{i=d}^{n} \underbrace{\frac{i-1}{i+1}}_{\leq 1} + c_1 \\
&\leq c_2 d! \sum_{i=d}^{n} constant \\
&\leq c_2 d! \cdot O(n) \\
&\leq O(d!n)
\end{aligned} \tag{1}$$

c) We need that with our choice of d, that:

$$O(d!n) \leq O(n^2)$$

This implies that we need:

$$O(d!) \leq O(n)$$

We choose:

$$d = \sqrt{\log n}$$

Since $d! \leq d^d$ and (with the De L'Hôpital rule) $\lim_{n\to\infty} \frac{\sqrt{\log n}^{\sqrt{\log n}}}{n} = 0$, we get that:

$$(\lfloor(\sqrt{\log n})\rfloor)! \leq \sqrt{\log n}^{\sqrt{\log n}} \leq n \qquad (\text{for } n \to \infty)$$

With that we get that with $d = \sqrt{\log n}$ we get an expected running time of $O(n^2)$

Now note that if we only choose $d = \log(n)$, we get a too big running time, because:
$\lim_{n\to\infty} \frac{(\log n)^{(\log n)}}{n} = \infty$.

Since $\sqrt{\log n} = (\log n)^{0.5}$ we get that the best possible value for d is between $(\log n)^{0.5}$ and $(\log n)^1$.

Let $d = (\log n)^\alpha$

We see that for all $0 \leq \alpha < 1$ we still get: $\lim_{n\to\infty} \frac{((\log n)^\alpha)^{((\log n)^\alpha)}}{n} = 0$
(This means that for $\alpha = 0.9999$ it still works, for a big enough n)

Therefore the upper limit would be $d = \log n$, however we would never exactly reach it.
We conclude that we will always have to choose:

$d = (\log n)^\alpha \qquad (\text{with } 0 \leq \alpha < 1).$