Algorithms, Probability and Computing: Special Assignment 1

Due 20 October 2015

Prof. Steger, Prof. Holenstein, Prof. Welzl

Kevin Klein

Exercise 1

Let P be the input set of points with $|P| = n$, with general position.

(1) - $\mathcal{O}(n \cdot log(n))$
Compute the convex hull of the set P. Runtime boundaries and correctnes can be guaranteed by using, for instance, Graham's algorithm.

(2) - $\mathcal{O}(n \cdot log(n))$
Compute the Voronoi diagram of the set P. Runtime boundaries and correctnes can be guaranteed by using, for instance, Fortune's algorithm. For every vertex in the Voronoi diagram, which are $\mathcal{O}(n)$ many [2], we query whether the vertex is inside of the convex hull within $\mathcal{O}(log(n))$ time.
If it is, we add the vertex to the queue Q.
If it is not, we check whether one of its edges is a 'ray', i.e. there is no second vertex on this edge. This is feasible in constant time. Thanks to the structure of the Voronoi diagram, we can look up the two closest points of P, $p_1$and p_2 for the current vertex. p_1 and p_2 have to be on the border of the convex hull. We follow all the edges from this vertex to other vertices until we reach a vertex inside the convex hull with every 'path'. For every such vertex, still outside of the convex hull, we check whether it has an edge which intersects with the edge e_{p_1,p_2}, i.e. the convex hull. If it does intersect, we add this intersection point to the queue Q, saving the points p_1 and p_2 with this intersection for determination of the radius. We can check whether two edges intersect in constant time. In order to avoid a superlinear amount of vertex-visits, we mark every node we visit when coming from a ray. Therefore we have $\mathcal{O}(n)$ vertices to visit and per vertex we only spend $\mathcal{O}(log(n))$ time.
After a one-time prepocessing of $\mathcal{O}(n)$ time, every query only takes $\mathcal{O}(log(n))$ time [0].

(3) - $\mathcal{O}(n)$
Iterate over all items in the queue. For each, check what the minimal distance to its closest point is. Save the maximum of the minimum distances to the neighbouring points, as well as the point where this distances is associated with.

(4) - $\mathcal{O}(1)$
Return this point associated with the maximum value, i.e. radius.

Soundness
If an point c is returned by the algorithm, it is either a vertex of the Voronoi diagram inside the convex hull or an intersection of a Voronoi diagram edge and the convex hull, i.e. a point lying on the relative interior of a segment of the convex hull.

a) c is vertex of Voronoi diagram
By the definition of step 2, we make sure that the vertex lies inside of the convex hull. $c \in conv(P)$ follows directly. The radius is adapted to the minimum of the distances to the neighbouring points of c, i.e. no point can be closer to c than the radius. Therefore $int(C) \cap P = \emptyset$.

b) c is intersection of Voronoi diagram and convex hull
As c is on the convex hull, $c \in conv(P)$ follows trivially. As described in step 3, when adding this point to the queue, we can determine its nearest points of P as the Voronoi diagram which points are closest to the edge, on which c is lying. Therefore we can determine minimum of the distances to those points and use it as radius. It follows from the correctness of the Voronoi diagram representation, that no point can be closer to c than the considered distance. Therefore $int(c) \cap P = \emptyset$.

Completeness
A point p we did not consider as a Posible solution is either lying inside the Voronoi cells or on an edge of the Voronoi diagram inside the convex hull, not intersection the convex hull.

a) p lies inside of a Voronoi cell
According to the definition of the Voronoi diagram, p is closer to a point of P than all the neighbouring edges and vertices of the Voronoi diagram. Transitivity of b) applies. Furthermore, those vertices/edges are not closer to any other point of P either, as they are defined to be equidistant to neighbouring points. Hence, no such point could yield a better solution.

b) p lies on an edge of the Voronoi diagram inside the convex hull, not intersection the convex hull

i) general case: the Voronoi cell, including all edges and vertices, is within the convex hull
At least one of the vertices of this cell is further away from the point associated with the cell than p. We considered this vertex as a Posibility of our solution. Therefore a point on the relative interior of an endge cannot yield a better solution.

ii) at least one Voronoi diagram vertex of a cell is outside of the convex hull
Compared to p, at least one vertex of the cell is further away from the point inside the cell. This vertex is either inside the convex hull or outside. If inside, we considered this vertex as a Posible solution. If outside, the furthest point from the point of the cell is necessarily another vertex or the intersection of the convex hull and an egde leading to the 'best' vertex, outside of the convex hull. This exact intersection has been considered by the algorithm as well. Hence no better solution could have been missed out.

[0] APC script, Chapter 2.1 Point/Line Relative to a Convex Polygon, paragraph 'Inside/On/Outside a Cnvex Polygon', p. 39 [1] APC script, Chapter 2.1 Point/Line Relative to a Convex Polygon, paragraph 'A Line Hitting a Convex Polygon', p. 40 [2] APC script, Chapter 2.3 Planar Point Location - More exmaples, paragrahp 'Closest Point in the Plane - the Post offce Problem', p.55

Exercise 2

(a) Let's say the blue points are indexed by $p_1 \dots p_i$ and the red points indexed bei $p_{i+1} \dots p_n$.

(1) - $\mathcal{O}(n)$
For every point $p = (a, b)$ generate the line $p^* = (a, -b)$. Chose an arbitrary red line and an arbitrary blue line and check which one is above. For the color being above, define the halfspace for every line of this color to point downwards. The lines of the opposite color point upwards.

(2) - $\mathcal{O}(n)$
Compute the intersection of those halfspaces.

(3) - $\mathcal{O}(1)$
If existent, return an the dual an arbitrary point of the intersection, i.e. one of the Posible intersecting lines.

Correctness
In order to prove correctness, it suffices to show that the problem that we want to solve and the we solve are equivalent.
Assume that if a seperation exists, the blue points are above, the opposite is analogous. We want to find a line l for which the following holds:
$l = (a, b), \forall p_k = (a_k, b_k) \in P$:

$$\begin{cases} b_k > a \cdot a_k + b & k \in \{1 \dots i\} \text{ i.e. every blue point lies above } l \\ b_k < a \cdot a_k + b & k \in \{i+1 \dots n\} \text{ i.e. every red point lies below } l \end{cases}$$

We can algebraically rewrite this as the following system of equations:

$$\begin{cases} -b > a_k * a - b_k & k \in \{1 \ldots i\} \text{ i.e. every blue line lies above point } l^* \\ -b < a_k * a - b_k & k \in \{i+1 \ldots n\} \text{ i.e. every red line lies below } l^* \end{cases}$$

This is equivalent to the statement, that every line $p_k^*, k \in \{1, \ldots, n\}$ has to be above the point l^* if red and below if blue. In order to determine the region in which such a point could be placed in, we can determine the intersection of the halfspaces - pointing upwards for the lines which have to be below and downwards in the other case. If the intersection is empty, there cannot be a point above all red lines and below all blue lines. If it is non-empty, any point inside of it fullfills the requirements. The concepts of duality allows us to retranslate this point to a line, which preserves the necessary properties.

(b) The line l lets us seperate the points into R_l and B_l. We can translate each of those sets into sets of lines, by the standard duality transformation. Lets say the sets contain i and j lines respectively. For now we consider R_l, as B_l is analogous. Furthermore we are aware of the fact that in each set, the number of lines is odd. Therefore, when searching for the $\lceil \frac{i}{2} \rceil$ level, we are looking for a set of points, which has $\lfloor \frac{i}{2} \rfloor$ lines above and $\lfloor \frac{i}{2} \rfloor$ lines below. It follows that this leads us to a curve.
When observing the curve from the left side of the leftmost intersection of our lines and the right side of the rightmost intersection of our lines, the curve is the same line, i.e. the line with the median slope. This is clear because without intersections, the k-level is trivially determined. In other words, the lines below the k-level on the right side will be above it on the right side and vice versa. It also follows that When pointing downwards on one extreme side, the k-level points upwards on the other extreme.
In the finite range, where intersections of lines happen, the k-level is not necessarily this line. However the k-level is still certain to be steady, according to the definition of the k-level.
Every statement applied to R_l applies to B_l.
As there is no pairwise equality of x-coordinates among the initial points, inexistence of pairwise equality of slopes follows for the lines, i.e. no two lines of R_l and B_l are parallel.
This means that when looking at the individual k-level of R_l and B_l combined, both have to intersect, as their k-level are apart from a finite range, non-parallel lines. Inside this finite range, they are still steady curves.
When converting this intersection back to the initial perspective of the duality, this point is equivalent to a line, which bisects the points of R_l and B_l as the intersection was a point which had $\lfloor \frac{i}{2} \rfloor$ lines of R_l above and below and $\lfloor \frac{|B_l|}{2} \rfloor$ lines of B_l above and below. This immediately from duality. Therefore a bisection must exist.
It suffices to consider the 'odd' case, as the 'even' or 'odd and even' cases are analogous apart from the fact that the k-levels don't need be curves, but are allowed to be areas.1

(c) *Preprocessing*
Given the points P, translate all of those to lines according to the standard dual transformation. Followingly determine the ranges from one line to another on the y-axis. This means that for any y, we can associate an interval, for which we know the first line above and first below.
As the input l could lie in any of those intervals, we consider every 'seperation Posibility' of P,i.e. we are prepared for every Posible seperation $B_l \cup R_l = P$.
For any such seperation we do the following:

- compute the k-levels of B_l
- compute the k-levels of R_l
- determine the intersections of the k-levels, if several intersections Posible, chose an arbitrary point inside
- save a mapping from the seperation-range to the intersection point in e.g. a hashtable

The second-last step can be implemented by naively comparing the intervals/piecewise curves as we explicitly do not care for the runtime of the preprocessing. Obviously, there are much faster solutions to this step.
Querying
When queried with a line $l : y = c$, all one has to do is to locate l, i.e. c in the seperation ranges. This works even thogh we changed the points into lines, as we only consider the x-axis which preservers the repartition properties. An adapted balanced binary search tree can assure us logarithmic query time. Instead of saving keys, the tree could simply save ranges as our ranges cannot overlap.
Correctness
The correctness of this algorithm follows directly from the definitions of the k-level, the properties of duality and the statements about the k-level-'line' from above.

Exercise 3

(a) $a_0 = 7, a_1 = 1 + 2 * a_0 = 15$
$\forall n \geq 2 :$

$$a_n = 1 + 2 \cdot \sum_{i=1}^{n} a_{i-1} \tag{1}$$
$$a_{n-1} = 1 + 2 \cdot \sum_{i=1}^{n} a_{i-1} \tag{2}$$
$$\Rightarrow a_n - a_{n-1} = 1 + 2 \cdot \sum_{i=1}^{n} a_{i-1} - (1 + 2 \cdot \sum_{i=1}^{n} a_{i-1}) = 2 \cdot a_{n-1} \tag{3}$$
$$\Rightarrow a_n = 3 \cdot a_{n-1} \tag{4}$$
$$= \dots \tag{5}$$
$$= 3^{n-1} \cdot a_1 \tag{6}$$
$$= 3^{n-1} \cdot 15 \tag{7}$$
$$\tag{8}$$

(b) $L_n := length\ of\ a\ central\ path\ in\ a\ BST\ of\ size\ n\ nodes$
$l_n := \mathbb{E}(L_n)$
$l_0 = 0, l_1 = 1, l_2 = 1/2 \cdot 2 + 1/2 \cdot 1 = 3/2$
$\forall n \geq 3 :$

$$l_n = \sum_{i=1}^{n} (\mathbb{E}[L_n | root = i] \cdot \Pr[root = i]) \tag{9}$$
$$= \sum_{i=1}^{n} (\mathbb{E}[L_n | root = i] \cdot 1/n) \tag{10}$$
$$= 1/n \cdot \sum_{i=1}^{n} (\mathbb{E}[L_n | root = i]) \tag{11}$$
$$= 1/n \cdot \sum_{i=1}^{n} (\mathbb{E}[L_{i-1}] + 1) \tag{12}$$
$$= 1 + 1/n \cdot \sum_{i=1}^{n} (l_{i-1}) \tag{13}$$
$$\tag{14}$$

Instead of recurring with two different expectations, one for going left and one for going right, we reduce the problem to always going left. In other words, this problem is equivalent to the expected length of the left spine. The legitimacy of this statement follows from the fact that we assume our binary search tree to be u.a.r. This tells us, that when 'arriving' at a node, we expect equal left and right subtrees, as they are random search trees themselves and the root was u.a.r. as well. In other words the decision which 'direction' we take is irrelevant when considering the expected length of the path.
Analogously, $l_{n-1} = 1/n \cdot \sum_{i=1}^{n-1}(l_{i-1})$.

$$\begin{aligned}
n \cdot l_n - (n-1) \cdot l_{n-1} &= n + \sum_{i=1}^{n}(l_{i-1}) - ((n-1) + \sum_{i=1}^{n-1}(l_{i-1}) && (15)\\
&= 1 + l_{n-1} && (16)\\
\Rightarrow n \cdot l_n &= 1 + n \cdot l_{n-1} && (17)\\
\Rightarrow l_n &= 1/n + l_{n-1} && (18)\\
&= 1/n + 1/(n-1) + \cdots + l_3 + l_2 && (19)\\
&= 1/n + 1/(n-1) + \cdots + l_3 + 3/2 && (20)\\
&= 1/n + 1/(n-1) + \cdots + l_3 + 1/2 + 1 && (21)\\
&= H_n && (22)\\
& && (23)
\end{aligned}$$

We see that this also holds for $n = 2$ and $n = 1$ and therefore: $\forall n \in \mathbb{N} - \{0\}$.

(c) $P_0^{(1)} = P_1^{(1)} = P_2^{(1)} = 0, P_3^{(1)} = 1/3, P_4^{(1)} = 1/4 \cdot 1/3 = 1/12$
$\forall n \geq 5 :$

$$\begin{aligned}
P_n^{(1)} &= \sum_{i=1}^{n}((P_n^{(1)}|root = i) \cdot \Pr[root = i]) && (24)\\
&= \sum_{i=1}^{n}((P_n^{(1)}|root = i) \cdot 1/n && (25)\\
&= 1/n \cdot \sum_{i=1}^{n}((P_n^{(1)}|root = i) && (26)\\
&= 1/n \cdot \sum_{i=1}^{n} P_{i-1}^{(1)} && (27)\\
& && (28)
\end{aligned}$$

Furthermore, $P_{n-1}^{(1)} = 1/(n-1) \cdot \sum_{i=1}^{n-1}((P_{i-1}^{(1)}|root = i)$

$$\begin{aligned}
\Rightarrow n \cdot P_n^{(1)} - (n-1) \cdot P_{n-1}^{(1)} &= \sum_{i=1}^{n} P_{i-1}^{(1)} - (\sum_{i=1}^{n-1} P_{i-1}^{(1)}) && (29)\\
&= P_{n-1}^{(1)} && (30)\\
\Rightarrow n \cdot P_n^{(1)} &= n \cdot P_{n-1}^{(1)} && (31)\\
\Rightarrow P_n^{(1)} &= P_{n-1}^{(1)} && (32)\\
&= P_{n-2}^{(1)} = \cdots = P_4^{(1)} && (33)\\
&= 1/12 && (34)
\end{aligned}$$

We see that this also holds for $n = 4$ and therefore: $\forall n \in \mathbb{N}, n \geq 4$.

(d) $n \geq 5,\ 2 \leq i \leq n-3$
As $i+1$ and $i+2$ are the only descendants of i, we have to make sure, that no other keys can end up children of i, $i+1$ and $i+2$. We can achieve this by requiring keys $i-1$ and $i+3$ to appear before i in the permutation of keys. This implies, that they are higher in the three, and therefore reached before i, $i+1$ and $i+2$. A key can then only land at i if and only if it is inside the range $i-1, \ldots, i+3$. As there are no keys in this range apart from the keys we care about, no other key than $i+1$ and $i+2$ can reach i. Therfore i will not have any other ancestors if this condition is satisfied.
Moreover, $i+1$ and $i+2$ have to follow i in the permutation for obvious reasons.

$$\begin{aligned}
\mathbb{E}[X_i] &= \sum x_i \cdot \Pr[X_i = x_i] && (35)\\
&= \Pr[X_i = 1] && (36)\\
&= \Pr[\text{keys } i-1 \text{ and } i+3 \text{ appear before } i,\ \text{keys } i+1 \text{ and } i+2 \text{ after } i] && (37)\\
& && (38)
\end{aligned}$$

As we consider u.a.r. trees, we can 'ignore' which and how many keys are between the keys we now care for.
How many permutations lead to an accepted tree? There are 2 accepted relative arrangements among $i-1$ and $i+3$, as it is irrelevant which of both comes first, as long as they preceed i. Thusly, i has a fixed relative position, i.e. only one Posible relative position. $i+1$ and $i+2$ also have two Posible arrangements, as their relative position is irrelevant, as long as they follow i.

$$\#AcceptedPermutations = 2 \cdot 1 \cdot 2 \ = 4$$

The amount of Posible permutations is obviously 5!.

$$\mathbb{E}[X_i] = \frac{4}{5!} = \frac{1}{30}$$

Exercise 4

(a) Let $S = \{1, \ldots, n\}$ be the set of all keys. Let $\Pr_S$ be the probability distribution on $\mathbb{D}_S$. For all $v \in V$, $w(v)$ denotes the number of keys in the subtree rooted at v.
We define $\widetilde{w}(v)$ followingly:

$$\widetilde{w}(v) = \begin{cases} 1 & w(v) \leq 2 \\ \binom{w(v)}{2} & otherwise \end{cases}$$

Let $t \in_{u.a.r.} \mathbb{D}_S$.
Accordingly, the probability of a tree with nodes $v \in V$ is the following:

$$\Pr_S[t] = \prod_{v \in V} \frac{1}{\widetilde{w}(v)}$$

Let's observe the different cases to confirm this hypothesis.
$n = 0:$
$t = \lambda \Rightarrow \Pr_S[t] = 1$
$n = 1:$
t is the tree with a single node and a single key $k \in n$ it its root. $\Rightarrow \Pr_S[t] = 1$
$n = 2:$
t is the tree with a single node and the keys $i, j \in n$ it its root. $\Rightarrow \Pr_S[t] = 1$

$n \geq 3$:
t has a root with keys $i, j \in n, i < j$. Furthermore, t has up to three subtrees t_l, t_m, t_r.

$$\Pr_S[t] = Pr_S[root = \{i, j\}] \cdot \Pr_S[t_l] \cdot \Pr_S[t_m] \cdot \Pr_S[t_r] \tag{39}$$
$$= \frac{1}{\binom{w(v)}{2}} \cdot \Pr_S[t_l] \cdot \Pr_S[t_m] \cdot \Pr_S[t_r] \tag{40}$$
$$= \frac{1}{\widetilde{w}(v)} \cdot \Pr_S[t_l] \cdot \Pr_S[t_m] \cdot \Pr_S[t_r] \tag{41}$$
$$\tag{42}$$

Analogously[3], 'we can conclude that the values suggested in the assertion of the observation satisfy the recurrence'. [3] APC script, Chapter 1.1 Definition, Lemma 1.1, p.4.

(b) In order to determine m_n and M_n we have a look at the 'best' and 'worst case' of our tree structure regarding the number of nodes.
Best case
Any construction holding two keys in each node is best regarding the use of nodes. If the number of keys happens to be odd, one node holds a single key. This leads immediately to the lower bound:

$$m_n = \left\lceil \frac{n}{2} \right\rceil$$

Proof
Let's assume this bound is not tight, i.e. it exists a tree of size n keys, containing $< \lceil \frac{n}{2} \rceil$ nodes. Hence this tree has at most$\lceil \frac{n}{2} \rceil - 1$ nodes, to which we assign n keys. The pigeonhole principle tells us that this will lead to at least one node to which 3 keys are assigned. This contradicts the definition of our tree strucutre and is therefore imPosible.
Worst case
Assuming $n = 4k + 1$.
A tree structure degenerating to a 'chain' is the least advantagous regarding the amount of required nodes per keys. By chain we mean to say that every node has three children, but only one of them contains two keys. Therefore, there are 4 keys per 3 nodes in a 'general' level of the tree. By gerneal we mean to say that first and last level are not to be considered at this stage of the argumentation. This leads to the upper bound:

$$M_n = \lfloor \frac{3n}{4} \rfloor + 1$$

Proof Let's assume this bound is not tight, i.e. it exists a tree of size n keys, containing $> \lfloor \frac{3n}{4} \rfloor + 1$ nodes. Hence this tree has at least$\lfloor \frac{3n}{4} \rfloor + 2 = 3k + 2$ nodes, to which we assign $n = 4k + 1$ keys.
The tree cannot grow anymore if not at least one of the nodes of a level, which comprises three nodes, has two keys. This holds because we can can 'count' the root for the last level and we also know that $n = 4k + 1$. In other words, $(3k + 2) \cdot \lfloor \frac{1}{3} \rfloor = k$ nodes must hold $(3k + 1) \cdot \lfloor \frac{1}{3} \rfloor \cdot 2 = 2k$ keys. Followingly, there are $3k + 2 - k = 2k + 2$ nodes and $4k + 1 - 2k = 2k + 1$ keys 'left'. Obviously, one node will not have a key, which contradicts our definition of a node. Therefore our assumption was wrong and our bound tight.

(c) Let's determine the probability a u.a.r. permutation of $[n]$ yields a u.a.r. ternary search tree T_n with n keys. Say this tree has keys i and j in its root with $i < j$.
In order to determine this probability, we look at the ratio of the amount of permutations which yield T_n and the amount of Posible permutations.
$Pos[T_i]$ = #valid permutations of T_i
$PPos[T_i^n]$ = #valid Posibilities to arrange the keys of T_i within n keys

$$
\begin{aligned}
Pos[T_n] &= Pos[root = \{i,j\}] \cdot Pos[T_{<i}] \cdot Pos[T_{>i,<j}] \cdot Pos[T_{>j}] \cdot PPos[T_{<i}^{n-2}] \cdot PPos[T_{>i,<j}^{n-i-1}] \cdot PPos[T_{>j}^{n-j}] && (43)\\
&= 2 \cdot Pos[T_{<i}] \cdot Pos[T_{>i,<j}] \cdot Pos[T_{>j}] \cdot \binom{(n-2)}{(i-1)} \cdot \binom{(n-i-1)}{(j-i-1)} \cdot \binom{(n-j)}{(n-j)} && (44)\\
&= 2 \cdot Pos[T_{<i}] \cdot Pos[T_{>i,<j}] \cdot Pos[T_{>j}] \cdot \binom{(n-2)}{(i-1)} \cdot \binom{(n-i-1)}{(j-i-1)} && (45)\\
\Pr[T_n] &= \frac{Pos[T_n]}{n!} && (46)\\
& && (47)
\end{aligned}
$$

(d) Let's prove the inequality for all n by inducing over subtrees.

Base cases
$\mathbb{E}[f(T_0)] = 0, \mathbb{E}[f(T_1)] = 1, \mathbb{E}[f(T_2)] = 1$
From $n = 1$ we see that $c \geq \frac{1}{3}$. Given this, the inequality clearly holds for the base cases.

Induction hypothesis
Assume that for an arbitrary $n \geq 3$:

$$\forall k < n : \mathbb{E}[f(T_k)] \leq \frac{2}{3}n + c$$

Induction step
As we consider $n \geq 3$, the root contains two roots i and j with $i < j$.

$$
\begin{aligned}
\mathbb{E}[f(T_n)] &= \sum_{j=1}^{n} \sum_{i=1}^{j-1} \mathbb{E}[f(T_n) | root = \{i,j\}] \cdot \Pr[root = \{i,j\}] && (48)\\
&= \Pr[root = \{i,j\}] \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} \mathbb{E}[f(T_{i-1})] + \mathbb{E}[f(T_{j-i-1})] + \mathbb{E}[f(T_{n-j})] + 1 && (49)\\
&= \frac{1}{\binom{n}{2}} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} \mathbb{E}[f(T_{i-1})] + \mathbb{E}[f(T_{j-i-1})] + \mathbb{E}[f(T_{n-j})] + 1 && (50)\\
&\leq \frac{1}{\binom{n}{2}} \sum_{j=1}^{n} \sum_{i=1}^{j-1} (\frac{2}{3}(i-1) + c) + (\frac{2}{3}(j-i-1) + c) + (\frac{2}{3}(n-j) + c) && (51)\\
&\leq \frac{1}{\binom{n}{2}} \sum_{j=1}^{n} \sum_{i=1}^{j-1} \frac{2}{3}i - \frac{2}{3}i + \frac{2}{3}j - \frac{2}{3}j + \frac{2}{3}n - \frac{2}{3} - \frac{2}{3} + 3c && (52)\\
&\leq \frac{1}{\binom{n}{2}} \sum_{j=1}^{n} \sum_{i=1}^{j-1} \frac{2}{3}n - \frac{4}{3} + 3c && (53)\\
&\leq \frac{1}{\binom{n}{2}} \frac{n \cdot (n-1)}{2} \cdot (\frac{2}{3}n - \frac{4}{3} + 3c) && (54)\\
&\leq \frac{2}{n \cdot (n-1)} \frac{n \cdot (n-1)}{2} \cdot (\frac{2}{3}n - \frac{4}{3} + 3c) && (55)\\
&\leq \frac{2}{3}n - \frac{4}{3} + 3c && (56)\\
& && (57)
\end{aligned}
$$

In order to validate the induction, we have to make sure that the initial claim holds, i.e. that the constant term stays the same.

$$c = -\frac{4}{3} + 3c \Rightarrow c = \frac{2}{3}$$

It remains to check, whether this is sound with the condition we recoginzed in the base case. It obviously does and therefore the induction holds.

(e) Suppose the assumption of (d) still holds, i.e. T_n is a u.a.r. ternary search tree. No matter what the definition of the search tree looks like, we can assume to that our root keys indicate the direction in which to go depending on the current key. No matter what this mapping looks like, as it is based on the search tree property, we can expect a u.a.r. distribution of keys onto the resepctive subtrees.
This allows us, for instance, to assume that within the sum of all Posible roots, we can expect every subtree to have equal expected properties.
For simplicity, assume that the set of keys is $[n]$.
$\mathbb{E}[f(T_0)] = 0, \mathbb{E}[f(T_1)] = 1, \mathbb{E}[f(T_2)] = 1$
$\forall n \leq 3$:

$$\mathbb{E}[f(T_n)] = \sum_{j=1}^{n}\sum_{i=1}^{j} \mathbb{E}[f(T_n) | root = \{i,j\}] \cdot \Pr[root = \{i,j\}] \tag{58}$$

$$= \Pr[root = \{i,j\}] \cdot \sum_{j=1}^{n}\sum_{i=1}^{j} \mathbb{E}[f(T_n) | root = \{i,j\}] \tag{59}$$

$$= \frac{1}{\binom{n}{2}} \cdot \sum_{j=1}^{n}\sum_{i=1}^{j} \mathbb{E}[f(T_l)] \cdot \mathbb{E}[f(T_m)] \cdot \mathbb{E}[f(T_r)] \tag{60}$$

$$= \frac{1}{\binom{n}{2}} \cdot \sum_{j=1}^{n}\sum_{i=1}^{j} \mathbb{E}[f(T_{<i})] \cdot \mathbb{E}[f(T_{>i,<j})] \cdot \mathbb{E}[f(T_{>j})] \tag{61}$$

$$= \frac{1}{\binom{n}{2}} \cdot \sum_{j=1}^{n}\sum_{i=1}^{j} 3 \cdot \mathbb{E}[f(T_{<i})] \tag{62}$$

$$= \frac{1}{\binom{n}{2}} \cdot \sum_{j=1}^{n}\sum_{i=1}^{j} 3 \cdot \mathbb{E}[f(T_{i-1})] \tag{63}$$

$$\mathbb{E}[f(T_{n-1})] = \frac{1}{\binom{(n-1)}{2}} \cdot \sum_{j=1}^{n-1}\sum_{i=1}^{j} 3 \cdot \mathbb{E}[f(T_{i-1})] \tag{64}$$

$$\binom{n}{2} \cdot \mathbb{E}[f(T_n)] - \binom{(n-1)}{2} \cdot \mathbb{E}[f(T_{n-1})] = \sum_{j=1}^{n}\sum_{i=1}^{j} 3 \cdot \mathbb{E}[f(T_{i-1})] - \sum_{j=1}^{n-1}\sum_{i=1}^{j} 3 \cdot \mathbb{E}[f(T_{i-1})] \tag{65}$$

$$= 3 \cdot \sum_{i=1}^{n} \mathbb{E}[f(T_{i-1})] \tag{66}$$

$$\Rightarrow \mathbb{E}[f(T_n)] = \frac{\binom{(n-1)}{2} + 1}{\binom{n}{2}} \cdot \mathbb{E}[f(T_{n-1})] + \frac{3}{\binom{n}{2}} \cdot \sum_{i=1}^{n-1} \mathbb{E}[f(T_{i-1})] \quad = \frac{\binom{(n-1)}{2} + 1}{\binom{n}{2}} \cdot \mathbb{E}[f(T_{n-1})] \tag{67}$$