CS 457, Fall 2019

Drexel University, Department of Computer Science

Lecture 6

Order Statistics and the Selection Problem

The i^{th} **order statistic** of a set of n numbers: the i^{th} smallest number in sorted sequence:

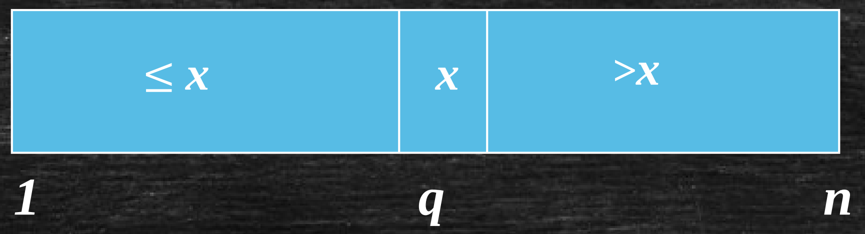
A	4	1	3	2	16	9	10	14	8	7

- Minimum or first order statistic: $\mathbf{1}$
- Maximum or n^{th} order statistic: $\mathbf{16}$
- Median or $(n/2)^{\text{th}}$ order statistic: $\mathbf{7}$ or $\mathbf{8}$ (both are medians, when n is even)

- **Selection Problem**
  - **Input:** An array $\mathbf{A}$ of distinct numbers of size n, and a number $\boldsymbol{i}$
  - **Output:** The element $\boldsymbol{x}$ in $\mathbf{A}$ that is larger than exactly $\boldsymbol{i}-\mathbf{1}$ other elements in $\mathbf{A}$
  - Finding ***maximum*** and ***minimum?***
  - Can be easily solved in linear time (i.e., $O(n)$. It's actually $\Theta(n)$)

Selection Algorithms using Pivot Element

- Choose a pivot element x and partition the subarray $A[1, \dots, n]$ around it



- If $q == i$, then x is the i^{th} order statistic
- If $q > i$, then we want the i^{th} order statistic of subarray $[1, \dots, q-1]$
- If $q < i$, then we want the $(i-q)^{\text{th}}$ order statistic of subarray $[q+1, \dots, n]$

- But, how do we choose this pivot element?

Simple Selection Algorithm

```
Select(A, p, r, i)

if p == r
    return A[p]
q = Partition(A, p, r)
k = q − p + 1
if i == k
    return A[q]
else if  i ≤ k
    Select(A, p, q − 1, i)
else
    Select(A, q + 1, r, i − k)
```

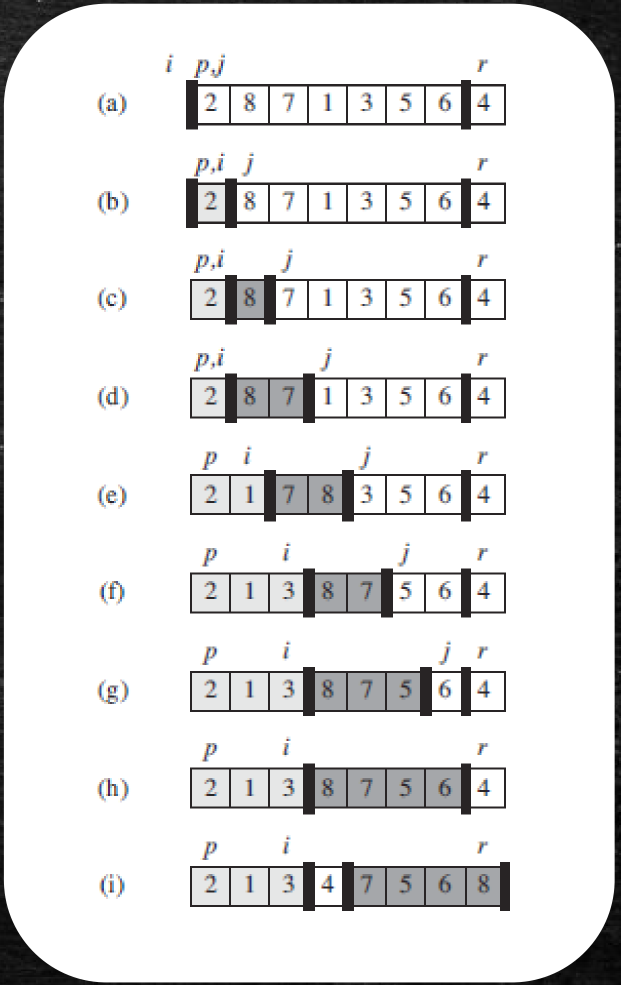
```
Partition (A, p, r)

x = A[r]
i=p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Partitioning

Partition (A, p, r)

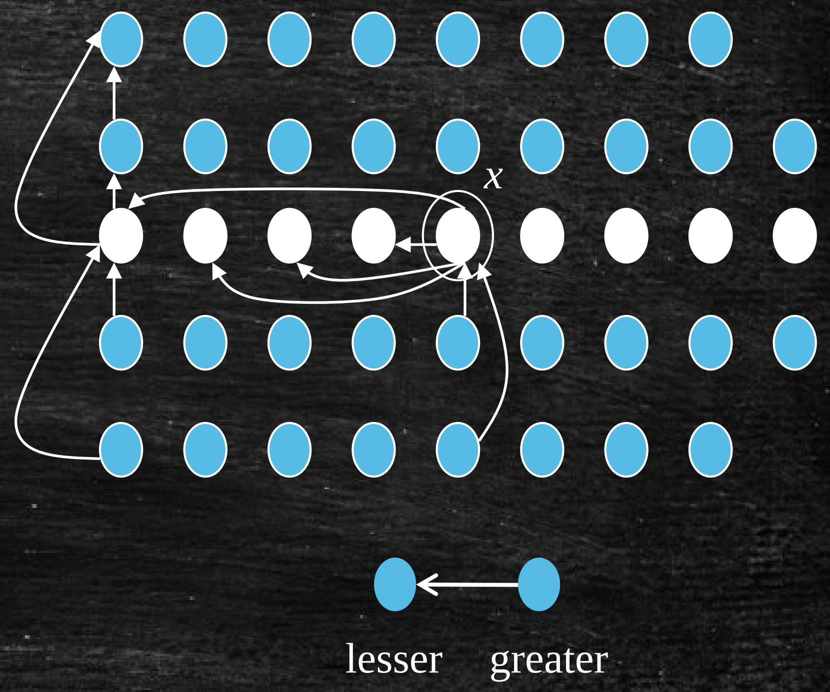
1. $x = A[r]$
2. $i = p - 1$
3. for $j = p$ to $r - 1$
4.   if $A[j] \leq x$
5. $i = i + 1$
6.   exchange $A[i]$ with $A[j]$
7. exchange $A[i + 1]$ with $A[r]$
8. return $i + 1$



Worst case linear time selection

Select(A,p,r,i)

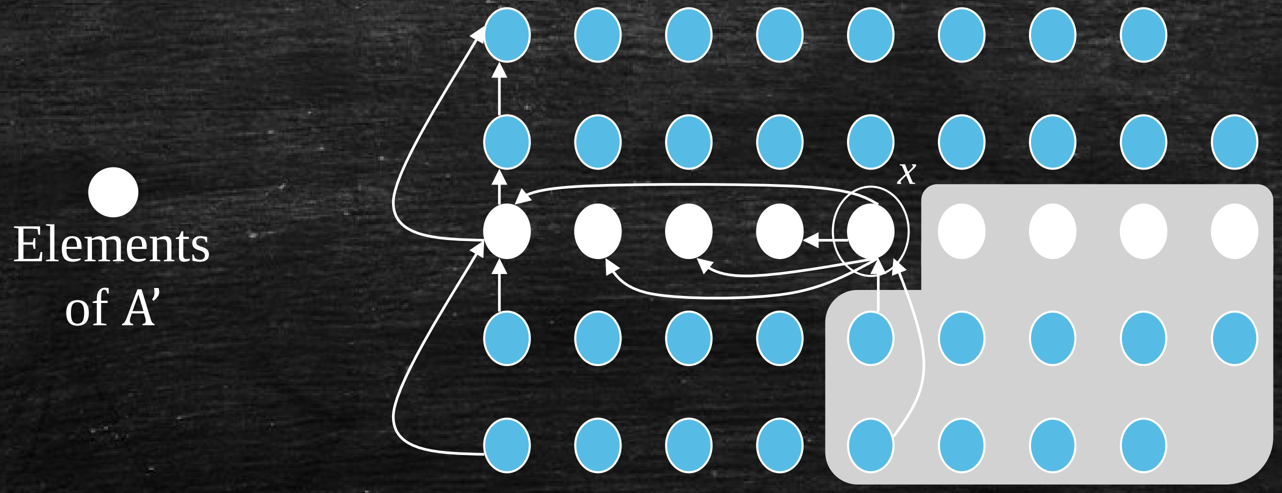
1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force, and store them in a set $\mathbf{A'}$ of size $n/5$.
3. Recursively use **Select**$(\mathbf{A'},\ 1,\ n/5, n/10)$ to find the median $\mathbf{x}$ of $n/5$ medians.
4. Partition elements of **A** around $\mathbf{x}$. Let k be the order of $\mathbf{x}$ found in the partitioning.
5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathbf{A},\ \mathbf{p},\ \mathbf{q} - 1,\ i)$
9. else
10. **Select**$(\mathbf{A}, \mathbf{q} + 1, \mathbf{r}, i - k)$



Today’s Lecture

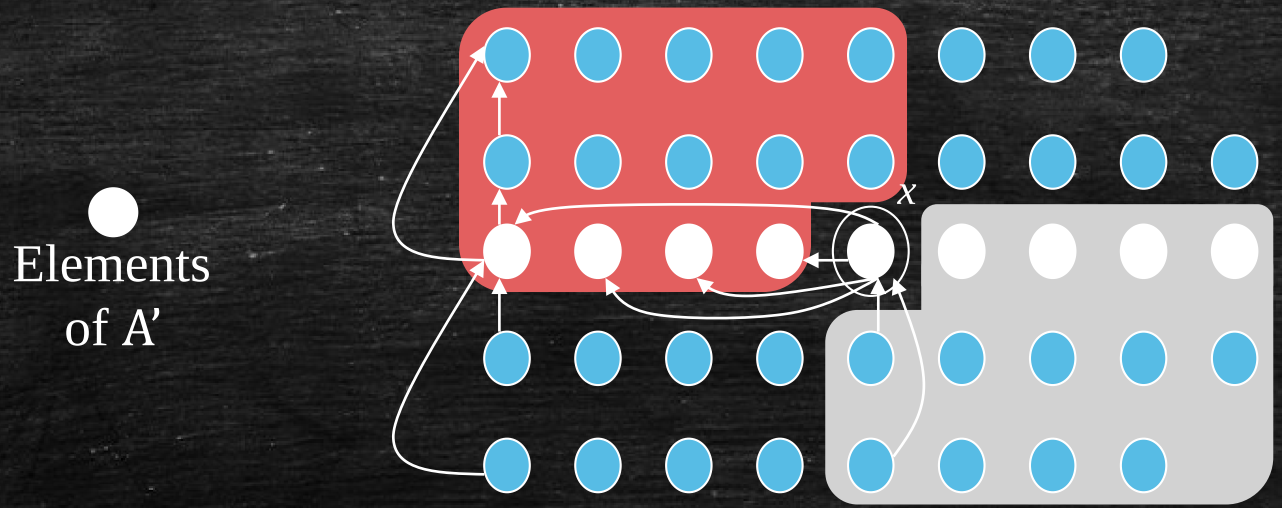
- More divide and conquer algorithms
- Probabilistic analysis and randomized algorithms

Analysis



- At least **half** of the $\lceil \boldsymbol{n}/\mathbf{5} \rceil$ elements in $\mathbf{A}'$ are $> \boldsymbol{x}$.
- Groups whose median $> \boldsymbol{x}$ have at least $\mathbf{3}$ elements $> \boldsymbol{x}$.
- Therefore, at least $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5}\right\rceil\right\rceil - 2\right) \geq \frac{3n}{10} - 6$ elements of $\boldsymbol{A}$ are $> \boldsymbol{x}$.

Analysis



- At least **half** of the $\lceil n/5 \rceil$ elements in $\mathbf{A}'$ are $< \boldsymbol{x}$.
- Groups whose median $< \boldsymbol{x}$ have at least $\mathbf{3}$ elements $< \boldsymbol{x}$.
- Therefore, at least $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$ elements of $\boldsymbol{A}$ are $< \boldsymbol{x}$.

Worst case linear time selection

Select(A,p,r,i)

1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force, and store them in a set $\mathbf{A}'$ of size $n/5$.
3. Recursively use **Select**$(\mathrm{A}', 1, n/5, n/10)$ to find the median $\mathbf{x}$ of $n/5$ medians.
4. Partition elements of **A** around $\mathbf{x}$. Let k be the order of $\mathbf{x}$ found in the partitioning.
5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathrm{A}, \mathrm{p}, \mathrm{q} - 1, i)$
9. else
10. **Select**$(\mathrm{A}, \mathrm{q} + 1, \mathrm{r}, i - k)$

Worst case linear time selection

Select(A,p,r,i)

1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force, and store them in a set $\mathbf{A}'$ of size $n/5$. $\Big\}\ O(n)$
3. Recursively use **Select**$(\mathrm{A}', \ 1, \ n/5, n/10)$ to find the median $\mathbf{x}$ of $n/5$ medians.
4. Partition elements of **A** around $\mathbf{x}$. Let k be the order of $\mathbf{x}$ found in the partitioning. $\Big\}\ O(n)$
5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathrm{A}, \ \mathrm{p}, \ \mathrm{q} - 1, \ i)$
9. else
10. **Select**$(\mathrm{A}, \mathrm{q} + 1, \mathrm{r}, i - k)$

Worst case linear time selection

Select(A,p,r,i)

1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force,
   and store them in a set $\mathbf{A'}$ of size $n/5$.

$O(n)$

3. Recursively use **Select**$(\mathrm{A'},\ 1,\ n/5, n/10)$
   to find the median $\mathbf{x}$ of $n/5$ medians.

$T\left(\left\lceil \frac{n}{5} \right\rceil\right)$

4. Partition elements of **A** around $\mathbf{x}$.
   Let k be the order of $\mathbf{x}$ found in the partitioning.

$O(n)$

5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathrm{A},\ \mathrm{p},\ \mathrm{q}-1,\ i)$
9. else
10. **Select**$(\mathrm{A}, \mathrm{q}+1, \mathrm{r}, i-k)$

Worst case linear time selection

Select(A,p,r,i)

1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force, and store them in a set $\mathbf{A}'$ of size $n/5$.

$O(n)$

3. Recursively use **Select**$(\mathrm{A}', \ 1, \ n/5, n/10)$ to find the median $\mathbf{x}$ of $n/5$ medians.

$T\left(\left\lceil\frac{n}{5}\right\rceil\right)$

4. Partition elements of **A** around $\mathbf{x}$. Let k be the order of $\mathbf{x}$ found in the partitioning.

$O(n)$

5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathrm{A}, \ \mathrm{p}, \ \mathrm{q} - 1, \ i)$
9. else
10. **Select**$(\mathrm{A}, \mathrm{q} + 1, \mathrm{r}, i - k)$

$T\left(\frac{7n}{10} + 6\right)$

Worst case linear time selection

Select(A,p,r,i)

1. Divide **A** into $n/5$ groups of size 5.
2. Find the median of each group of 5 by brute force, and store them in a set $\mathbf{A'}$ of size $n/5$.

$O(n)$

3. Recursively use **Select**$(\mathrm{A'},\ 1,\ n/5, n/10)$ to find the median $\mathbf{x}$ of $n/5$ medians.

$T\left(\left\lceil \frac{n}{5} \right\rceil\right)$

4. Partition elements of **A** around $\mathbf{x}$. Let k be the order of $\mathbf{x}$ found in the partitioning.

$O(n)$

5. if $i = k$
6. return $\mathbf{x}$
7. else if $i < k$
8. **Select**$(\mathrm{A},\ \mathrm{p},\ \mathrm{q}-1,\ i)$
9. else
10. **Select**$(\mathrm{A}, \mathrm{q}+1, \mathrm{r}, i-k)$

$T\left(\frac{7n}{10}+6\right)$

$$T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10}+6\right) + O(n)$$

Worst case linear time selection

Guess that $T(n) = O(n)$ when $T(n) \leq T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + O(n)$

Assume $T(n') \leq cn'$ for some constant c and every $n' < n$, and

show that this implies $T(n) \leq cn$ (for the same constant c)

Upon substitution, we get $T(n) \leq c\left\lceil \frac{n}{5} \right\rceil + c\left(\frac{7n}{10} + 6\right) + O(n)$

For $n > n_0$ this is at most $c\left(\frac{n}{5} + 1\right) + c\left(\frac{7n}{10} + 6\right) + an$

Therefore, $T(n) \leq cn - \left(\frac{cn}{10} - 7c - an\right)$

Worst case linear time selection

Guess that $T(n) = O(n)$ when $T(n) \leq T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + O(n)$

Assume $T(n') \leq cn'$ for some constant c and every $n' < n$, and

show that this implies $T(n) \leq cn$ (for the same constant c)

Upon substitution, we get $T(n) \leq c\left\lceil \frac{n}{5} \right\rceil + c\left(\frac{7n}{10} + 6\right) + O(n)$

For $n > n_0$ this is at most $c\left(\frac{n}{5} + 1\right) + c\left(\frac{7n}{10}\right.$ …

for $c \geq 20a$
and $n \geq 140.$

Therefore, $T(n) \leq cn - \left(\frac{cn}{10} - 7c - an\right) \leq cn$

Simple Selection Algorithm

```
Select(A, p, r, i)

if p == r
    return A[p]
q = Partition(A, p, r)
k = q − p + 1
if i == k
    return A[q]
else if  i ≤ k
    Select(A,  p,  q − 1,  i)
else
    Select(A,  q + 1,  r,  i − k)
```

```
Partition (A, p, r)

x = A[r]
i=p − 1
for j = p to r − 1
    if  A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Randomized Selection Algorithm

Randomized-Select(A, p, r, i)

if $p == r$
    return $A[p]$
q = Randomized-Partition(A, p, r)
$k = q - p + 1$
if $i == k$
    return $A[q]$
else if $i \leq k$
    Randomized-Select$(A,\ p,\ q-1,\ i)$
else
    Randomized-Select$(A,\ q+1,\ r,\ i-k)$

Randomized-Partition (A, p, r)

i = Random(p, r)
Exchange $A[r]$ with $A[i]$
return Partition(A, p, r)

Randomized Selection Algorithm

Randomized-Select(A, p, r, i)

```
if p == r
    return A[p]
q = Randomized-Partition(A, p, r)
k = q − p + 1
if i == k
    return A[q]
else if i ≤ k
    Randomized-Select(A, p, q − 1, i)
else
    Randomized-Select(A, q + 1, r, i − k)
```

- Worst-case running time?
  - a) $O(n^2)$
  - b) $O(n \log n)$
  - c) $O(n)$
- Expected running time?
  - a) $O(n^2)$
  - b) $O(n \log n)$
  - c) $O(n)$

Running Time

- Indicator variable $\mathbb{I}\{E\}$ is 1 if event E occurs and 0 o/w (see page 118)
- Consider an array $A[p, \dots, r]$ with n elements
- If $X_k = \mathbb{I}\{$the subarray $A[p, \dots, q]$ has exactly k elements$\}$, then

$$T(n) \leq \sum_{k=1}^{n} X_k \left(T(\max\{k-1, n-k\}) + O(n)\right)$$

$$= \sum_{k=1}^{n} X_k \left(T(\max\{k-1, n-k\})\right) + \sum_{k=1}^{n} X_k \, O(n)$$

$$= \sum_{k=1}^{n} X_k \left(T(\max\{k-1, n-k\})\right) + O(n)$$

Expected Running Time

- $X_k = \mathbb{I}\{$the subarray $A[p, \ldots, q]$ has exactly k elements$\}$. Since $A[p, \ldots, r]$ has n elements and q is chosen uniformly at random, we have $\mathbb{E}[X_k] = \frac{1}{n}$, so

$$
\begin{aligned}
\mathbb{E}[T(n)] &\le \mathbb{E}\left[\sum_{k=1}^{n} X_k \left(T(\max\{k-1, n-k\})\right) + O(n)\right] \\
&= \sum_{k=1}^{n} \mathbb{E}[X_k (T(\max\{k-1, n-k\}))] + O(n) \\
&= \sum_{k=1}^{n} \mathbb{E}[X_k]\, \mathbb{E}[T(\max\{k-1, n-k\})] + O(n) \\
&= \sum_{k=1}^{n} \frac{1}{n}\, \mathbb{E}[T(\max\{k-1, n-k\})] + O(n)
\end{aligned}
$$

Expected Running Time

Thus, $\mathbb{E}[T(n)] \leq \sum_{k=1}^{n} \frac{1}{n}\, \mathbb{E}[T(\max\{k-1, n-k\})] + O(n),$

and $\max\{k-1, n-k\}) = \begin{cases} k-1 \text{ if } k > \lceil n/2 \rceil \\ n-k \text{ if } k \leq \lceil n/2 \rceil \end{cases}$

This is a recurrence
that we can now solve

So, $\mathbb{E}[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} \mathbb{E}[T(k)] + O(n)$

Expected Running Time

We use substitution in order to solve: $\mathbb{E}[T(n)] \leq \frac{2}{n} \sum_{k=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} \mathbb{E}[T(k)] + O(n)$

We guess that $\mathbb{E}[T(n)] = O(n)$

Hence, we assume $\mathbb{E}[T(n')] \leq cn'$ for some constant c and every $n' < n$,

and we show that $\mathbb{E}[T(n)] \leq cn$ for that same constant c. Upon substitution:

$$\mathbb{E}[T(n)] \leq \frac{2}{n} \sum_{k=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} ck + an \leq \frac{2c}{n}\left(\frac{n^2-n}{2} - \frac{n^2/4 - 3n/2 + 2}{2}\right) + an$$

which leads to $\mathbb{E}[T(n)] \leq cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right)$

Expected Running Time

We use substitution in order to solve: $\mathbb{E}[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} \mathbb{E}[T(k)] + O(n)$

We guess that $\mathbb{E}[T(n)] = O(n)$

Hence, we assume $\mathbb{E}[T(n')] \leq cn'$ for some constant c and every $n' < n$,

and we show that $\mathbb{E}[T(n)] \leq cn$ for that same constant c. Upon substitution:

$$\mathbb{E}[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} ck + an \leq \frac{2c}{n}\left(\frac{n^2 - n}{2} - \cdots\right) + an$$

for $c > 4a$,
and $n \geq \frac{2c}{c-4a}$

which leads to $\mathbb{E}[T(n)] \leq cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right) \leq cn$