IFT436 - Revue des techniques de preuve et applications en analyse d'algorithmes

Preuve directe

Type de preuve avec séquence de déductions.

Preuve directe

Type de preuve avec séquence de déductions.

Théorème

Si x et y sont des entiers pairs, alors $x + y$ est un entier pair.

Preuve directe

Type de preuve avec séquence de déductions.

Théorème

Si x et y sont des entiers pairs, alors $x + y$ est un entier pair.

Preuve. Puisque x et y sont pairs, il existe des entiers k et h tels que $x = 2k$ et $y = 2h$. On a

$$x + y = 2k + 2h = 2(k + h),$$

ce qui implique que $x + y$ est pair.

Preuve par contre-exemple

Pour démontrer un énoncé du style $\neg(\forall x \cdot A)$, il suffit de trouver un seul exemple de x qui rend A faux.

Preuve par contre-exemple

Pour démontrer un énoncé du style $\neg(\forall x \cdot A)$, il suffit de trouver un seul exemple de x qui rend A faux.

Théorème

Le tri sous-gradué présenté en classe ne fonctionne pas correctement.

Preuve par contre-exemple

Pour démontrer un énoncé du style $\neg(\forall x \cdot A)$, il suffit de trouver un seul exemple de x qui rend A faux.

Théorème

Le tri sous-gradué présenté en classe ne fonctionne pas correctement.

$\neg(\forall$ tableau $T \cdot ($ l'algo trie $T)))$

Preuve par contre-exemple

Pour démontrer un énoncé du style $\neg(\forall x \cdot A)$, il suffit de trouver un seul exemple de x qui rend A faux.

Théorème

Le tri sous-gradué présenté en classe ne fonctionne pas correctement.

$\neg(\forall$ tableau $T \cdot ($ l'algo trie $T)))$

Preuve. Sur entrée $[3, 1, 2, 4]$, à la première itération, quand $i = 1$, l'algorithme verra que $T[i] > T[i + 1]$, car $3 > 1$. Il va donc échanger $T[1]$ avec $T[n - nbswap] = T[4]$. Donc, 3 se retrouvera à la dernière position. Ensuite, puisque i est toujours plus petit que n, et que $n - nbswap$ est plus petit que n pour le reste de l'exécution, ce 3 en fin de tableau ne sera jamais remplacé. Le tableau retourné ne sera donc pas trié.

Preuve par contradiction

Pour prouver A, on suppose $\neg A$ et on dérive quelque de faux ou d'impossible.

Voir la preuve que $\sqrt{2}$ est un nombre irrationnel. On suppose que $\sqrt{2} = \frac{a}{b}$, tels que a et b n'ont pas de multiple commun autre que 1, et on dérive qu'en fait a et b sont pairs.

Preuve par induction

Pour montrer que $P(n)$ est vrai pour tout $n \geq a$:
1. Cas de base : Montrer que $P(a)$ est vrai.
2. Induction : Montrer pour tout $n > a$ que si $P(n-1)$ est vrai, alors $P(n)$ est vrai.
(ou de façon équivalente, pour tout $n \geq a$, si $P(n)$ est vrai alors $P(n+1)$ est vrai)

Preuve par induction

Pour montrer que $P(n)$ est vrai pour tout $n \geq a$:
1. Cas de base : Montrer que $P(a)$ est vrai.
2. Induction : Montrer pour tout $n > a$ que si $P(n-1)$ est vrai, alors $P(n)$ est vrai.
(ou de façon équivalente, pour tout $n \geq a$, si $P(n)$ est vrai alors $P(n+1)$ est vrai)

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

On a

$$\sum_{i=0}^{n} 2^i =$$

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

On a

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

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

On a

$$\begin{aligned} \sum_{i=0}^{n} 2^i &= \sum_{i=0}^{n-1} 2^i + 2^n \\ &= 2^n - 1 + 2^n \quad \text{par hypothèse d'induction} \end{aligned}$$

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

On a

$$\begin{aligned} \sum_{i=0}^{n} 2^i &= \sum_{i=0}^{n-1} 2^i + 2^n \\ &= 2^n - 1 + 2^n \quad \text{par hypothèse d'induction} \\ &= 2 \cdot 2^n - 1 \end{aligned}$$

Preuve par induction

Théorème

Pour tout $n \geq 0$, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Cas de base : $n = 0$. $\sum_{i=0}^{0} 2^i = 2^0 = 1$. Aussi, $2^{0+1} - 1 = 1$. Donc, l'égalité est vraie.

Induction : on suppose que $\sum_{i=0}^{n-1} 2^i = 2^{n-1+1} - 1 = 2^n - 1$ est vrai. On veut montrer $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

On a

$$\begin{aligned}
\sum_{i=0}^{n} 2^i &= \sum_{i=0}^{n-1} 2^i + 2^n \\
&= 2^n - 1 + 2^n \quad \text{par hypothèse d'induction} \\
&= 2 \cdot 2^n - 1 \\
&= 2^{n+1} - 1.
\end{aligned}$$

Preuve par induction

Parfois il faut plusieurs cas de base.
Pour montrer que $P(n)$ est vrai pour tout $n \geq a$:
1. Cas de base : Montrer que $P(a), P(a+1), \ldots, P(a+b)$ sont vrais.
2. Induction : Montrer pour tout $n > b$ que si $P(n-1)$ est vrai, alors $P(n)$ est vrai.

Preuve par induction

Parfois il faut plusieurs cas de base.
Pour montrer que $P(n)$ est vrai pour tout $n \geq a$:
1. Cas de base : Montrer que $P(a), P(a+1), \ldots, P(a+b)$ sont vrais.
2. Induction : Montrer pour tout $n > b$ que si $P(n-1)$ est vrai, alors $P(n)$ est vrai.

Aussi correct : en 2. Induction, montrer pour tout $n \geq b$ que si $P(n)$ est vrai, alors $P(n+1)$ est vrai.

Induction forte

Pour montrer que $P(n)$ est vrai pour tout $n \geq a$:
1. Cas de base : Montrer que $P(a), P(a+1), \ldots, P(a+b)$ sont vrais.
2. Induction : Montrer pour tout $n > b$ que si $P(a), P(a+1), \ldots, P(n-1)$ sont vrais, alors $P(n)$ est vrai.

Induction forte

Théorème

Tout nombre $n \in \mathbb{N}$ tel que $n \geq 2$ est le produit de un ou plusieurs nombres premiers.

Induction forte

Théorème

Tout nombre $n \in \mathbb{N}$ tel que $n \geq 2$ est le produit de un ou plusieurs nombres premiers.

Cas de base. Si $n = 2$, n est divisible par 2, qui est un nombre premier (et donc le produit d'un nomre premier).
Induction. Pour $n > 2$, on suppose que tout nombre entre 2 et $n - 1$ est le produit de nombres premiers. On veut prouver que n est le produit de nombres premiers.
Si n est déjà premier, il n'y a rien à prouver. Sinon, n est un nombre composé et donc $n = p \cdot q$. Ici, $2 \leq p < n$ et $2 \leq q < n$ doivent être vrais, et par induction, p et q sont le produit de nombres premiers, i.e., $p = p_1 p_2 \dots p_a$ et $q = q_1 q_2 \dots q_b$, où les p_i et q_i sont premiers. On a $n = pq = p_1 \dots p_a q_1 \dots q_b$, qui est un produit de nombres premiers.

Fouille dichotomique récursive

Entrée : tableau **trié** A, élément x
Sortie : *vrai* si x est dans A, *faux* sinon.

Algorithme 1 : Fouille dichotomique

Fonction `FouilleDicho`(A, x) **:**

```
n = |A|;
si n = 0 alors retourner faux;
si n = 1 alors retourner A[1] == x;
m = n/2 + 1;
si A[m] = x alors
    retourner vrai;
sinon si x < A[m] alors
    retourner FouilleDicho(A[1, ..., m − 1], x);
sinon
    retourner FouilleDicho(A[m + 1, ..., n], x);
```

Fouille dichotomique récursive

Théorème

Si A est un tableau trié, alors FouilleDicho fonctionne correctement. C'est-à-dire, il retourne *vrai* si x est dans A, et *faux* sinon.

Fouille dichotomique récursive

Théorème

Si A est un tableau trié, alors FouilleDicho fonctionne correctement. C'est-à-dire, il retourne *vrai* si x est dans A, et *faux* sinon.

Par induction sur $n = |A|$.
Cas de base : $n = 0$ et $n = 1$.
Si $n = 0$, retourner *faux* est correct car A ne contient pas x.
Si $n = 1$, on aura $m = n/2 + 1$ qui prendra la valeur de 1. Si A contient x, c'est à la position 1. On aura $A[1] = x$ et on retourne correctement *vrai*. Si A ne contient pas x, $A[1] \neq x$ et on retournera *faux*.

Fouille dichotomique récursive

Induction : on suppose que FouilleDicho est correct sur tout tableau de taille entre 0 et $n-1$. On veut montrer qu'il est correct sur A de taille n.
Si $A[m] == x$, alors on retourne *vrai* et c'est évidemment correct.

Fouille dichotomique récursive

Induction : on suppose que FouilleDicho est correct sur tout tableau de taille entre 0 et $n-1$. On veut montrer qu'il est correct sur A de taille n.
Si $A[m] == x$, alors on retourne *vrai* et c'est évidemment correct.

Si $x < A[m]$, alors on appelle FouilleDicho$(A[1..m-1], x)$, qui par hypothèse d'induction retourne *vrai* si et seulement si x est dans le sous-tableau. Si x est dans A, alors x est forcément dans $A[1..m-1]$ car $x < A[m]$, dans quel case l'appel récursif retournera *vrai* et nous aussi. Si x n'est pas dans A, il n'est pas dans le sous-tableau, l'appel récursif retournera *faux* et nous aussi.

Fouille dichotomique récursive

Induction : on suppose que FouilleDicho est correct sur tout tableau de taille entre 0 et $n-1$. On veut montrer qu'il est correct sur A de taille n.
Si $A[m] == x$, alors on retourne *vrai* et c'est évidemment correct.

Si $x < A[m]$, alors on appelle FouilleDicho$(A[1..m-1], x)$, qui par hypothèse d'induction retourne *vrai* si et seulement si x est dans le sous-tableau. Si x est dans A, alors x est forcément dans $A[1..m-1]$ car $x < A[m]$, dans quel case l'appel récursif retournera *vrai* et nous aussi. Si x n'est pas dans A, il n'est pas dans le sous-tableau, l'appel récursif retournera *faux* et nous aussi.

Si $x > A[m]$, l'argument est le même : on appelle FouilleDicho$(A[m+1, \ldots, n], x)$, et si x est dans A il est dans le sous-tableau $A[m+1, \ldots, n]$ et, par induction, l'appel récursif retournera une réponse correcte et nous aussi.

Fouille dichotomique récursive

Théorème

Si A est un tableau de taille $n \geq 1$, FouilleDicho fait au plus $\log n + 1$ appels récursifs (en comptant l'appel initial).

Fouille dichotomique récursive

Théorème

Si A est un tableau de taille $n \geq 1$, FouilleDicho fait au plus $\log n + 1$ appels récursifs (en comptant l'appel initial).

Par induction sur n.

Cas de base : si $n = 1$, on fait un seul appel et $\log 1 + 1 = 0 + 1 = 1$.

Fouille dichotomique récursive

Induction : on suppose que pour tout tableau de taille k entre 1 et $n-1$, FouilleDicho fait au plus $\log k$ appels récursifs.

Fouille dichotomique récursive

Induction : on suppose que pour tout tableau de taille k entre 1 et $n-1$, FouilleDicho fait au plus $\log k$ appels récursifs.
Supposons que l'appel récursif se fait sur $A[1, \ldots, m-1]$. Si n est pair, alors $m = n/2 + 1$ et la taille du sous-tableau est $n/2$.

Fouille dichotomique récursive

Induction : on suppose que pour tout tableau de taille k entre 1 et $n-1$, FouilleDicho fait au plus $\log k$ appels récursifs.
Supposons que l'appel récursif se fait sur $A[1, \ldots, m-1]$. Si n est pair, alors $m = n/2 + 1$ et la taille du sous-tableau est $n/2$. Par hypothèse d'induction, cet appel fait au plus $\log(n/2) + 1$ appels. En ajoutant l'appel courant, il y en a $\log(n/2) + 2$.

Fouille dichotomique récursive

Induction : on suppose que pour tout tableau de taille k entre 1 et $n-1$, FouilleDicho fait au plus $\log k$ appels récursifs.
Supposons que l'appel récursif se fait sur $A[1,\ldots,m-1]$. Si n est pair, alors $m = n/2+1$ et la taille du sous-tableau est $n/2$. Par hypothèse d'induction, cet appel fait au plus $\log(n/2)+1$ appels. En ajoutant l'appel courant, il y en a $\log(n/2)+2$. On a

$$\log(n/2)+2 = \log n - \log 2 + 2 = \log n - 1 + 2 = \log n + 1.$$

Fouille dichotomique récursive

Induction : on suppose que pour tout tableau de taille k entre 1 et $n-1$, FouilleDicho fait au plus $\log k$ appels récursifs.
Supposons que l'appel récursif se fait sur $A[1,\ldots,m-1]$. Si n est pair, alors $m = n/2+1$ et la taille du sous-tableau est $n/2$. Par hypothèse d'induction, cet appel fait au plus $\log(n/2)+1$ appels. En ajoutant l'appel courant, il y en a $\log(n/2)+2$. On a

$$\log(n/2)+2 = \log n - \log 2 + 2 = \log n - 1 + 2 = \log n + 1.$$

Si n est impair, alors $n/2$ sera tronqué, et en fait $m = (n-1)/2+1$. La taille du sous-tableau est donc $(n-1)/2$. Par induction, l'appel fait au plus $\log((n-1)/2)+1$ appels, donc $\log((n-1)/2)+2$ au total. On a

$$\log((n-1)/2)+2 = \log(n-1) - \log 2 + 2 = \log(n-1)+1 \leq \log n + 1.$$

En exercice : le cas où l'appel se fait sur $A[m+1,\ldots,n]$.

Analyse du tri fusion

Théorème

Le Tri Fusion trie correctement.

Analyse du tri fusion

Théorème

Le Tri Fusion trie correctement.

Lemme. Soit A et B deux tableaux triés. Alors la boucle de fusion produit un tableau S trié contenant les éléments de A et B. (en exercice)

Analyse du tri fusion

Théorème

Le Tri Fusion trie correctement.

Lemme. Soit A et B deux tableaux triés. Alors la boucle de fusion produit un tableau S trié contenant les éléments de A et B. (en exercice)

Preuve du théorème. On prouve que le Tri Fusion est correct par induction sur n.
Cas de base : si $n = 1$, le tableau retourné est clairement trié.

Analyse du tri fusion

Théorème

Le Tri Fusion trie correctement.

Lemme. Soit A et B deux tableaux triés. Alors la boucle de fusion produit un tableau S trié contenant les éléments de A et B. (en exercice)

Preuve du théorème. On prouve que le Tri Fusion est correct par induction sur n.
Cas de base : si $n = 1$, le tableau retourné est clairement trié.

Induction : on suppose que Tri Fusion trie tout tableau dont la taille est entre 1 et $n - 1$. On montre qu'il est correct pour n.
Par hypothèse d'induction, le tableau A contient une version triée de $T[1, \ldots, n/2]$, et par hypothèse d'induction, B contient une version triée de $T[n/2 + 1, \ldots, n]$.

Analyse du tri fusion

Théorème

Le Tri Fusion trie correctement.

Lemme. Soit A et B deux tableaux triés. Alors la boucle de fusion produit un tableau S trié contenant les éléments de A et B. (en exercice)

Preuve du théorème. On prouve que le Tri Fusion est correct par induction sur n.
Cas de base : si $n = 1$, le tableau retourné est clairement trié.

Induction : on suppose que Tri Fusion trie tout tableau dont la taille est entre 1 et $n - 1$. On montre qu'il est correct pour n.
Par hypothèse d'induction, le tableau A contient une version triée de $T[1, \ldots, n/2]$, et par hypothèse d'induction, B contient une version triée de $T[n/2 + 1, \ldots, n]$.

Par le lemme, le S retourné produit une version triée des éléments de A et B, et donc une version triée de T (car A et B contiennent exactement les éléments de T).

Analyse du tri fusion

Théorème

Pour $n = 2^k$, où $k \geq 0$, le Tri Fusion fait au plus $n \log n$ comparaisons.
(une comparaison compare un élément de A avec celui de B)

On prend n comme puissance de 2 pour éviter d'avoir à gérer les cas pair/impair.

Analyse du tri fusion

Théorème

Pour $n = 2^k$, où $k \geq 0$, le Tri Fusion fait au plus $n \log n$ comparaisons.
(une comparaison compare un élément de A avec celui de B)

On prend n comme puissance de 2 pour éviter d'avoir à gérer les cas pair/impair.

Par induction sur la puissance k.
Cas de base. Si $k = 0$, alors $n = 2^0 = 1$. Dans ce cas, l'algo fait 0 comparaison. Ceci est correct car $n \log n = 1 \cdot \log 1 = 1 \cdot 0 = 0$.

Analyse du tri fusion

Induction. on suppose que sur un tableau de taille 2^{k-1}, alors Tri Fusion fait au plus $2^{k-1} \log 2^{k-1}$ comparaisons. On veut prouver l'énoncé pour $n = 2^k$.

Soit $c(n)$ le nombre de comparaisons faites par l'algorithme sur un tableau de taille n.

Analyse du tri fusion

Induction. on suppose que sur un tableau de taille 2^{k-1}, alors Tri Fusion fait au plus $2^{k-1} \log 2^{k-1}$ comparaisons. On veut prouver l'énoncé pour $n = 2^k$.

Soit $c(n)$ le nombre de comparaisons faites par l'algorithme sur un tableau de taille n.

Remarquons que $2^{k-1} = 2^k/2 = n/2$. Donc l'hypothèse d'induction est que $c(n/2) \leq n/2 \cdot \log(n/2)$.

Analyse du tri fusion

Induction. on suppose que sur un tableau de taille 2^{k-1}, alors Tri Fusion fait au plus $2^{k-1} \log 2^{k-1}$ comparaisons. On veut prouver l'énoncé pour $n = 2^k$.

Soit $c(n)$ le nombre de comparaisons faites par l'algorithme sur un tableau de taille n.

Remarquons que $2^{k-1} = 2^k/2 = n/2$. Donc l'hypothèse d'induction est que $c(n/2) \leq n/2 \cdot \log(n/2)$.

Sur entrée de taille $n > 1$, chaque appel récursif fait $c(n/2)$ comparaisons, et il y a 2 appels. Ensuite, on voit que la boucle de fusion ne fait jamais plus que n comparaisons.

Analyse du tri fusion

Induction. on suppose que sur un tableau de taille 2^{k-1}, alors Tri Fusion fait au plus $2^{k-1} \log 2^{k-1}$ comparaisons. On veut prouver l'énoncé pour $n = 2^k$.

Soit $c(n)$ le nombre de comparaisons faites par l'algorithme sur un tableau de taille n.

Remarquons que $2^{k-1} = 2^k/2 = n/2$. Donc l'hypothèse d'induction est que $c(n/2) \leq n/2 \cdot \log(n/2)$.

Sur entrée de taille $n > 1$, chaque appel récursif fait $c(n/2)$ comparaisons, et il y a 2 appels. Ensuite, on voit que la boucle de fusion ne fait jamais plus que n comparaisons.

Donc, $c(n) \leq 2 \cdot c(n/2) + n$

Analyse du tri fusion

On a :

$$c(n) \leq 2 \cdot c(n/2) + n$$

Analyse du tri fusion

On a :

$$\begin{aligned} c(n) &\leq 2 \cdot c(n/2) + n \\ &\leq 2 \cdot n/2 \cdot \log(n/2) + n \quad \text{(par hypothèse d'induction)} \end{aligned}$$

Analyse du tri fusion

On a :

$$
\begin{aligned}
c(n) &\leq 2 \cdot c(n/2) + n \\
&\leq 2 \cdot n/2 \cdot \log(n/2) + n \quad \text{(par hypothèse d'induction)} \\
&= n \cdot (\log n - \log 2) + n
\end{aligned}
$$

Analyse du tri fusion

On a :

$$
\begin{aligned}
c(n) &\leq 2 \cdot c(n/2) + n \\
&\leq 2 \cdot n/2 \cdot \log(n/2) + n \quad \text{(par hypothèse d'induction)} \\
&= n \cdot (\log n - \log 2) + n \\
&= n \cdot (\log n - 1) + n
\end{aligned}
$$

Analyse du tri fusion

On a :

$$
\begin{aligned}
c(n) &\leq 2 \cdot c(n/2) + n \\
&\leq 2 \cdot n/2 \cdot \log(n/2) + n \quad \text{(par hypothèse d'induction)} \\
&= n \cdot (\log n - \log 2) + n \\
&= n \cdot (\log n - 1) + n \\
&= n \log n - n + n
\end{aligned}
$$

Analyse du tri fusion

On a :

$$
\begin{aligned}
c(n) &\leq 2 \cdot c(n/2) + n \\
&\leq 2 \cdot n/2 \cdot \log(n/2) + n \quad \text{(par hypothèse d'induction)} \\
&= n \cdot (\log n - \log 2) + n \\
&= n \cdot (\log n - 1) + n \\
&= n \log n - n + n \\
&= n \log n.
\end{aligned}
$$