1

c

$$
\begin{aligned}
T(n) &= 2T(\lceil n/4 \rceil) + n \\
\frac{n}{4^i} &= 1 \\
i &= \log_4 n \\
num\_levels &= depth + 1 = \log_4 n + 1 \\
num\_nodes(depth = i) &= 2^i \\
cost\_per\_node(depth = i) &= \frac{n}{4^i} \\
cost\_per\_level(depth = i) &= num\_nodes(depth = i) \cdot cost\_per\_node(depth = i) \\
&= 2^i \cdot \frac{n}{4^i} \\
&= \left(\frac{2}{4}\right)^i n \\
&= \left(\frac{1}{2}\right)^i n \\
&= \frac{n}{2^i} \\
T(n) &= \sum_{i=0}^{max\_depth} cost\_per\_level(depth = i) \\
&= \sum_{i=0}^{\log_4 n} \frac{n}{2^i} \\
&= n \sum_{i=0}^{\log_4 n} \frac{1}{2^i} \\
&= n \left(\frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + + ... + \frac{1}{2^{\log_4 n}} \right) \\
&= \Theta(n)
\end{aligned}
$$

2
c

```
/* suppose the elements of a are sorted.
you are given an array a[1...n]
of length n consisting of some positive
and some negative integers in sorted order.
design a divide and conquer-based algorithm
to find the first positive integer in a.
your algorithm must work in O(log n) time.

example: If a = {-7, -3, 6, 8, 11, 12},
then the first positive integer listed in a is a[3] = 6,
so the algorithm must return 6 */

#include <stdio.h>

int find(int * a, int left, int right)
{/* find i s.t. a[i - 1] < 0, a[i] > 0 */
    if(left > right)
        return 1 << 31;
    int i = (left + right) / 2;
    if(!i && a[i] > 0)
        return a[i];
    else if(a[i - 1] < 0 && a[i] > 0)
        return a[i];
    else if(a[i] > 0)
        return find(a, left, i - 1);
    else if(a[i] < 0)
        return find(a, i + 1, right);
}


int main()
{
    int a[] = {-7, -3, 6, 8, 11, 12};
    int n = sizeof(a) / sizeof(*a);
    printf("%d\n", find(a, 0, n - 1));
```

```
}
```

3

a

```
#include <stdio.h>
#include <math.h>

int choose_box_rec(int * a, int n, int k)
{
    if(n - 1 < 0)
        return 0;
    return fmax(choose_box_rec(a, n - 1, k),
    a[n - 1] + choose_box_rec(a, n - k - 1, k));
}

int choose_box(int * a, int n, int k)
{
    int memo[n];
    for(int i = 0; i < n; ++i)
    {
        int a1 = 0, a2 = 0;
        if(i - 1 >= 0)
            a1 = memo[i - 1];
        if(i - k - 1 >= 0)
            a2 = memo[i - k - 1];
        memo[i] = fmax(a1, a2 + a[i]);
    }
    return memo[n - 1];
}

int main()
{
    int a[] = {-7, -3, 6, 8, 11, 12};
```

```
    int n = sizeof(a) / sizeof(*a);
    int k = 2;
    printf("choose_box_rec = %d\n", choose_box_rec(a, n, k));
    printf("choose_box = %d\n", choose_box(a, n, k));
}
```