```
#include <iostream>
using namespace std;

int maxsimum(int *A, int l, int r){
	if (l == r) return A[l];

	int m = (l+r)/2;
	int left = maxsimum(A,l,m); //logn
	int right = maxsimum(A,m+1,r); //logn

	return left > right ? left : right;
}

int main(){
	int A[] = {5,6,7,8,9};
	int n = sizeof(A)/sizeof(int);
	cout <<"Maximum is: " << maxsimum(A,0,n-1) << endl;

	return 0;
}
```

n is the size of the list

Space complexity

陣列 A 需要 n 個空間，其他變數各 1 個空間，共 c 個空間。

O(n+c) = n

Time complexity

Best case:

當只有一個元素時，時間複雜度為 O(1)

Wosrt case

$log_2^n + C = log_2^n + C$

$O(log_2^n + C) = log_2^n$