[illegible]

[illegible] OF ENGINEERING & [illegible]

SCIENCES [illegible] MATHEMATICAL

```
int sumOfCubes(int n) {
    int partialSum = 0;
    for (int i = 1; i <= n; ++i) {
        partialSum += i * i * i;
    }
    return partialSum;
}
```

```
int sumOfCubes(int n) {
    int partialSum = 0;
    for (int i = 1; i <= n; ++i) {
        partialSum += i * i * i;
    }
    return partialSum;
}
```

$O\,n$

```cpp
int twoDArraySum(std::vector<std::vector<int>> a) {
    int result = 0;
    for (int i = 0; i < a.size(); i++) {
        std::vector<int> row = a[i];
        for (int j = 0; j < a[i].size(); j++) {
            result = result + row[j];
        }
    }
    return result;
}
```

```
int twoDArraySum(std::vector<std::vector<int>> a) {
    int result = 0;
    for (int i = 0; i < a.size(); i++) {
        std::vector<int> row = a[i];
        for (int j = 0; j < a[i].size(); j++) {
```

```
int maxSubSum1(const std::vector<int> &a) {
    int maxSum = 0;
    for (int i = 0; i < a.size(); ++i) {  // left boundary
        for (int j = i; j < a.size(); ++j) {  // right boundary
            int thisSum = 0;
            for (int k = i; k <= j; ++k) {
                thisSum += a[k];
            }
            if (thisSum > maxSum) {
                maxSum = thisSum;
            }
        }
    }
    return maxSum;
}
```

```
int maxSubSum1(const std::vector<int> &a) {
    int maxSum = 0;
    for (int i = 0; i < a.size(); ++i) {  // left boundary
        for (int j = i; j < a.size(); ++j) {  // right boundary
            int thisSum = 0;
            for (int k = i; k <= j; ++k) {
                thisSum += a[k];
            }
            if (thisSum > maxSum) {
                maxSum = thisSum;
            }
        }
    }
    return maxSum;
}
```

$O\ n^3$

```
int maxSubSum2(const std::vector<int> &a) {
```

```
int maxSubSum2(const std::vector<int> &a) {
    int maxSum = 0;
    for (int i = 0; i < a.size(); ++i) {
        int thisSum = 0;
        for (int j = i; j < a.size(); ++j) {
            thisSum += a[j];
            if (thisSum > maxSum) {
                maxSum = thisSum;
            }
        }
    }
    return maxSum;
}
```

$O\ n^2$

```
int maxSumRec(const std::vector<int> &a, int left, int right) {
    if (left == right) {
        if (a[left] > 0) {
            return a[left];
        } else {
            return 0;
        }
    }
    int center = (left + right) / 2; // divide into two halves
    int maxLeftSum = maxSumRec(a, left, center);  // recursive call
    int maxRightSum = maxSumRec(a, center + 1, right);  // recursive call
    int maxLeftBorderSum = 0, leftBorderSum = 0;
    for (int i = center; i >= left; --i) {
        leftBorderSum += a[i];
        if (leftBorderSum > maxLeftBorderSum) {
            maxLeftBorderSum = leftBorderSum;
        }
```

```c
int sum(int a, int b, int c) {
    int sum = a + b + c;
    return sum;
}
```

```
int sum(int a, int b, int c) {
    int sum = a + b + c;
    return sum;
}
```

O

O

O

```cpp
int sum(std::vector<int> nums) {
    int sum = 0;
    for (int i = 0; i < nums.size(); ++i){
        sum += nums[i];
    }
    return sum;
}
```

```
int sum(std::vector<int> nums) {
    int sum = 0;
    for (int i = 0; i < nums.size(); ++i){
        sum += nums[i];
    }
    return sum;
}
```

O

O

O

ff

ff

e.g

O N K O

```
vector<vector<int>> mult(vector<int> nums1, vector<int> nums2) {
    vector<vector<int>> product;
    product.resize(nums1.size());
```

```
nums1 = {1, 2, 3, 4}
nums2 = {5, 0, 2}
    product
```

```
vector<vector<int>> mult(vector<int> nums1, vector<int> nums2) {
    vector<vector<int>> product;
    product.resize(nums1.size());
    for (int i = 0; i < nums1.size(); ++i) {
        product[i].resize(nums2.size());
        for (int j = 0; j < nums2.size(); ++j) {
            product[i][j] = nums1[i] * nums2[j];
        }
    }
    return product;
}
```

$O \quad \times$

$O \quad \times$

$O \quad \times$