|  |  |
| --- | --- |
| **TRƯỜNG TH, THCS VÀ THPT HOÀNG VIỆT**  TỔ CÔNG NGHỆ THÔNG TIN | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  Độc lập – Tự do – Hạnh phúc |

**HƯỚNG DẪN CHẤM KIỂM TRA NĂNG LỰC HỌC SINH GIỎI KHỐI 11**

**LẦN 4**

**Bài 1: Tổng chữ số**

**Cách giải:**

* Duyệt qua từng chữ số của **N**, cộng dồn vào biến tổng.
* Thời gian chạy: **O(log N)**.

**Mã C++:**

|  |
| --- |
| #include <iostream>  using namespace std;  int sumOfDigits(int n) {  int sum = 0;  while (n > 0) {  sum += n % 10;  n /= 10;  }  return sum;  }  int main() {  int n;  cin >> n;  cout << sumOfDigits(n) << endl;  return 0;  } |

**Bộ test:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 12345 | 15 |
| 9876 | 30 |
| 5 | 5 |
| 1001 | 2 |
| 99999 | 45 |
| 10 | 1 |
| 123456789 | 45 |
| 1111 | 4 |
| 54321 | 15 |
| 77777 | 35 |

**Bài 2: Ước số nguyên tố lớn nhất**

**Cách giải:**

* Chia hết số **N** cho **2** nhiều lần (nếu có).
* Lặp qua các số lẻ từ **3** đến **√N**, chia nhiều lần nếu **N** chia hết.
* Nếu **N** còn lại > 1, nó chính là ước số nguyên tố lớn nhất.
* Thời gian chạy: **O(√N)**.

**Mã C++:**

|  |
| --- |
| #include <iostream>  using namespace std;  long long largestPrimeFactor(long long n) {  long long maxPrime = -1;    while (n % 2 == 0) {  maxPrime = 2;  n /= 2;  }  for (long long i = 3; i \* i <= n; i += 2) {  while (n % i == 0) {  maxPrime = i;  n /= i;  }  }  if (n > 1) maxPrime = n;    return maxPrime;  }  int main() {  long long n;  cin >> n;  cout << largestPrimeFactor(n) << endl;  return 0;  } |

**Bộ test:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10 | 5 |
| 13195 | 29 |
| 97 | 97 |
| 100 | 5 |
| 77 | 11 |
| 99991 | 99991 |
| 9999991 | 757 |
| 27 | 3 |
| 81 | 3 |
| 18 | 3 |

**Bài 3: Dãy con có tổng lớn nhất (Kadane)**

**Cách giải:**

* Dùng thuật toán **Kadane** để tính tổng con lớn nhất.
* Thời gian chạy: **O(N)**.

**Mã C++:**

|  |
| --- |
| #include <iostream>  #include <vector>  #include <climits>  using namespace std;  int maxSubArray(vector<int> &arr) {  int maxSum = INT\_MIN, curSum = 0;  for (int x : arr) {  curSum = max(x, curSum + x);  maxSum = max(maxSum, curSum);  }  return maxSum;  }  int main() {  int n;  cin >> n;  vector<int> arr(n);  for (int &x : arr) cin >> x;  cout << maxSubArray(arr) << endl;  return 0;  } |

**Bộ test:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 -2 1 -3 4 -1 2 1 -5 4 | 6 |
| 3 -1 -2 -3 | -1 |
| 6 1 2 3 4 5 -10 | 15 |
| 4 10 -2 -3 4 | 10 |
| 7 -5 1 2 3 4 -10 | 10 |
| 5 -1 -1 -1 -1 -1 | -1 |
| 3 0 0 0 | 0 |
| 10 1 -2 3 -1 2 -3 4 -1 5 -6 | 7 |
| 4 10 -3 4 2 | 13 |
| 5 2 -1 2 3 -9 | 5 |

**Bài 4: Biểu thức ngoặc đúng (Stack)**

**Cách giải:**

* Dùng Stack kiểm tra tính hợp lệ của chuỗi.
* Thời gian chạy: **O(N)**.

**Mã C++:**

|  |
| --- |
| #include <iostream>  #include <stack>  using namespace std;  bool isValidParentheses(string s) {  stack<char> stk;  for (char c : s) {  if (c == '(') stk.push(c);  else {  if (stk.empty()) return false;  stk.pop();  }  }  return stk.empty();  }  int main() {  string s;  cin >> s;  cout << (isValidParentheses(s) ? "YES" : "NO") << endl;  return 0;  } |

**Bộ test:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| (()) | YES |
| ()() | YES |
| (() | NO |
| ))(( | NO |
| ()()() | YES |
| ((())) | YES |
| ()((())()) | YES |
| ((( | NO |
| ))) | NO |
| ()(()()) | YES |

**Bài 5: Đường đi ngắn nhất (Dijkstra)**

**Cách giải:**

* Dùng thuật toán **Dijkstra** để tìm đường đi ngắn nhất.
* Thời gian chạy: **O((N+M) log N)**.

**Mã C++:**

|  |
| --- |
| #include <iostream>  #include <vector>  #include <queue>  #include <limits.h>  using namespace std;  struct Edge {  int to, weight;  };  vector<long long> dijkstra(int n, vector<vector<Edge>> &adj) {  vector<long long> dist(n + 1, LLONG\_MAX);  priority\_queue<pair<long long, int>, vector<pair<long long, int>>, greater<>> pq;    dist[1] = 0;  pq.push({0, 1});  while (!pq.empty()) {  auto [d, u] = pq.top(); pq.pop();  if (d > dist[u]) continue;  for (auto e : adj[u]) {  if (dist[e.to] > dist[u] + e.weight) {  dist[e.to] = dist[u] + e.weight;  pq.push({dist[e.to], e.to});  }  }  }  return dist;  }  int main() {  int n, m;  cin >> n >> m;  vector<vector<Edge>> adj(n + 1);  for (int i = 0; i < m; i++) {  int u, v, w;  cin >> u >> v >> w;  adj[u].push\_back({v, w});  adj[v].push\_back({u, w});  }  auto dist = dijkstra(n, adj);  cout << (dist[n] == LLONG\_MAX ? -1 : dist[n]) << endl;  return 0;  } |

**Bộ test:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 1  1 2 5 | 5 |
| 5 4  1 2 2  1 3 3  1 4 4  1 5 5 | 5 |
| 5 4  1 2 1  2 3 1  3 4 1  4 5 1 | 4 |
| 4 5  1 2 1  2 3 2  3 4 3  1 3 2  2 4 4 | 4 |
| 3 1  1 2 5 | -1 |
| 4 4  1 2 1  2 3 1  3 4 1  4 2 1 | 3 |
| 3 3  1 2 1000000  2 3 1000000  1 3 2000000 | 2000000 |
| 6 9  1 2 1  2 3 2  3 4 3  4 5 4  5 6 5  1 3 2  2 4 3  3 5 4  4 6 5 | 6 |
| 5 6  1 2 2  2 3 2  3 4 2  4 5 2  1 5 10  3 5 1 | 7 |
| 6 8  1 2 1  2 3 1  3 6 1  1 4 2  4 5 2  5 6 2  2 5 2  3 4 2 | 4 |