AtCoder・パソコン甲子園でよく使う応用的なC++コード集

1. ソート・二分探索

```
vector<int> a = {1, 5, 3, 2, 4};
sort(a.begin(), a.end()); // 昇順に並べ替え
reverse(a.begin(), a.end()); // 降順に並べ替え

int x = 3;
bool found = binary_search(a.begin(), a.end(), x); // xが存在するか判定
```

sortで配列を整列し、binary_searchで要素が存在するかを高速に判定できます。

2. 集合とマップ

```
set<int> s;
s.insert(3);
s.insert(1);
s.insert(2); // s = {1, 2, 3}

map<string, int> mp;
mp["apple"] = 5;
mp["banana"] = 2;
```

setは重複を許さない集合、mapはキーと値の対応付けを管理できます。

3. 優先度付きキュー（ヒープ）

```
priority_queue<int> pq; // 最大ヒープ
pq.push(3);
pq.push(10);
pq.push(5);
cout << pq.top() << endl; // 10

priority_queue<int, vector<int>, greater<int>> pq_min; // 最小ヒープ
```

最大値や最小値を効率よく取り出すときに使います。

4. 組み合わせの全探索 (bit全探索)

```
int n = 3;
for (int bit = 0; bit < (1<<n); ++bit) {
  vector<int> subset;
  for (int i = 0; i < n; ++i) {
    if (bit & (1<<i)) subset.push_back(i);
  }
  // subsetに選ばれた要素が入る
}
```

n要素の部分集合をすべて探索する方法です。

5. 累積和

```
vector<int> a = {2, 1, 3, 4, 5};
int n = a.size();
vector<int> prefix(n+1, 0);
for (int i = 0; i < n; i++) prefix[i+1] = prefix[i] + a[i];

// 区間 [l, r) の和 = prefix[r] - prefix[l]
```

累積和を使うと、任意の区間和をO(1)で計算できます。

6. BFS・DFS（グラフ探索）

```
// BFS（幅優先探索）
queue<int> q;
vector<int> dist(n, -1);
dist[0] = 0;
q.push(0);
while (!q.empty()) {
    int v = q.front(); q.pop();
    for (int u : graph[v]) {
        if (dist[u] == -1) {
            dist[u] = dist[v] + 1;
            q.push(u);
        }
    }
}
```

BFSは最短距離の計算に使われます。DFSは再帰やスタックで探索します。

7. 動的計画法（DP）

```
int n = 5;
vector<int> dp(n+1, 1e9);
dp[0] = 0;
for (int i = 0; i < n; i++) {
    dp[i+1] = min(dp[i+1], dp[i] + 1);
    if (i+2 <= n) dp[i+2] = min(dp[i+2], dp[i] + 1);
}
```

DPは最小コストや最適化問題を解くときに必須です。

8. 高速入出力

```
ostringstream oss;
for (int i = 0; i < n; i++) {
    oss << a[i] << "\n";
}
cout << oss.str();
```

大量出力を一度にまとめることで、高速化できます。