London

Klejdi Sevdari

November 14, 2025

Problem restatement

For the purpose of solving the problem, we may view it in a simplified way: the note is a string of characters that must be assembled by reordering the characters obtainable from all front–back pairs of the newspaper. Since each piece can contribute exactly one of its two letters, the task reduces to checking whether the multiset of characters in the note is contained in the multiset of characters available from the newspaper pieces.

More formally, if the front letter is f and the back letter (mirrored) is b, then the piece can contribute either f or b to the note. It cannot be used twice.

Flow Construction

We construct a directed graph with

- a source node s,
- a sink node t,
- 26 intermediate nodes, one for each letter $A, \ldots, Z$.

Let $v(c)$ denote the node corresponding to the letter c. We first count the number of occurrences of each character in the note. For every letter c, we then add an edge $(v(c), t)$ with capacity equal to its frequency in the note. Similarly, we count how many times each character appears on the front side of the newspaper and add edges $(s, v(c))$ with capacities equal to these frequencies (see figure 1).

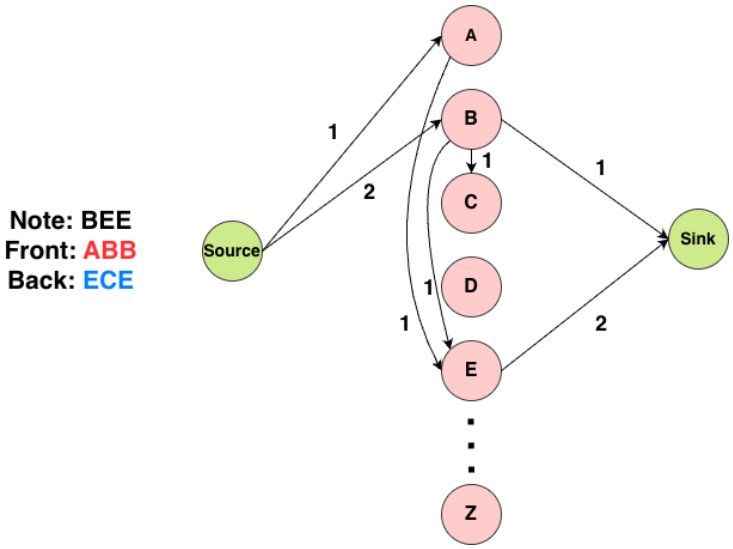

Figure 1: Simple example of flow construction.

In addition, each newspaper piece has a front letter f and a corresponding back letter b. Since a piece can be flipped, it may contribute either f or b, but never both. To model this choice, for each such pair we add an edge $(v(f), v(b))$ with capacity 1. This allows one unit of flow originating from a front letter to be redirected to its back letter if the piece is used in flipped orientation. If a certain pair occurs multiple times, we increase the capacity of $(v(f), v(b))$ accordingly.

In this construction, every unit of flow sent from s corresponds to selecting a newspaper piece, passing through either its front letter or its back letter, and finally contributing that letter to satisfy the demand at the sink. Computing the maximum s–t flow therefore determines whether all characters required by the note can be supplied by the available newspaper pieces.

Decision Criterion

We compute the maximum s–t flow. Let n be the length of the note. If the max-flow equals n, then each character demand is satisfied by some piece, respecting the front/back choices and ensuring no piece is used more than once.

If the flow is strictly less than n, the note cannot be constructed.

Code implementation

```
// init graph
graph G(26);
const vertex_desc v_source = boost::add_vertex(G);
const vertex_desc v_sink = boost::add_vertex(G);
edge_adder adder(G);

// add edges
for (auto c : note) {
   adder.increment_capacity(c - 'A', v_sink);
}

for (int i = 0; i < h * w; i++) {
   int f = front[i] - 'A', b = back[i] - 'A';
   adder.increment_capacity(v_source, f);
   adder.increment_capacity(f, b);
}

// solve
long flow = boost::push_relabel_max_flow(G, v_source, v_sink);

// output
if(flow==note.size()){
cout << "Yes\n";
}else{
cout << "No\n";
}
```