Zadanie M - BFS

Twoim zadaniem jest zaimplementowanie funkcji *make_bfs* - funkcji budującej funkcję przeszukiwania grafu wszerz. Funkcja otrzymuje dwa argumenty:

- funkcję, która dla danego wierzchołka zwraca parę iteratorów, wyznaczających listę sąsiedztwa tego wierzchołka
- funkcję, która ma zostać wywołana na wierzchołku gdy jest on po raz pierwszy odwiedzany

Zwrócona funkcja przyjmuje dwa argumenty - parę iteratorów, wyznaczających zbiór elementów inicjalizujących kolejkę BFS-a. Funkcja wykonuje algorytm BFS i zwraca liczbę odwiedzonych wierzchołków.

Co ważne: graf jest zaimplementowany jako *vector<int>* (tj. wierzchołki są typu *int*, a listy sąsiedztwa typu *vector<int>*).

Uwagi:

- rozwiązanie powinno znajdować się w pliku *bfs.h*
- złożoność algorytmu BFS nie może być kwadratowa!
- należy wykorzystywać możliwie mało pamięci

Zadanie wymaga użycia C++11, kompilacja odbywa się w standardzie C++14. **Należy skorzystać z interfejsu funkcyjnego. Nie trzeba korzystać z szablonów.**

Przykład:

Dla poniższego kodu

```
#include <cstdio>
#include <vector>
#include "bfs.h"


int main(){
    int N = 10;
    std::vector<int> vec;
    for(int i=0;i<N;i++) vec.push_back(i);
    auto bfs = make_bfs(
        [&vec,N] (int x){return std::make_pair(std::next(vec.begin(),x+1),std::next(vec.begin(),x+1<N
        [](int x) {printf("Action on node %d\n",x);}
    );
    int num = bfs(vec.begin(), vec.begin()+1);
    printf("------\n");
    printf("NUM of actions = %d\n", num);
}
```

Wynikiem jest

```
Action on node 0
Action on node 1
Action on node 2
Action on node 3
Action on node 4
Action on node 5
Action on node 6
Action on node 7
Action on node 8
Action on node 9
------
NUM of actions = 10
```