public int[] pairingAlgorithm(){  
 int n = this.getNodeCount();  
 int x;  
 PeopleNode vPeopleNode, xPeopleNode;  
   
 int[] matching = new int[n]; // tablica skojarzen  
  
 int[] alternatingPath = new int[n]; // n elementowa pomocnicza tablica służąca do tworzenia ścieżki rozszerzającej. Przechowuje tworzoną przez BFS strukturę drzewa rozpinającego wszerz, gdzie i-ty element zawiera numer swojego wierzchołka nadrządnego.  
  
 Boolean[] visited = new Boolean[n]; // pomocnicza tablica logiczna służąca do zaznaczania odwiedzonych wierzcholkow  
  
 LinkedList<Integer> queue = new LinkedList<>(); // kolejka, w ktorej sa skladowane wierzcholki dla BFS  
  
 for(int i = 0 ; i < n ; i++){  
 matching[i] = (-1);  
 }

long startTime = System.*nanoTime*();  
 //zasadniczy algorytm  
 for(int v = 0; v < n; v++){  
 vPeopleNode = (PeopleNode) this.getNode(v);  
 //sprawdzamy czy dany wierzcholek jest kobieta i nie jest skojarzona z mezczyzna, pomijamy wierzchołki skojarzone oraz kawalerów  
 if(matching[v] == (-1) && vPeopleNode.getPerson().getSex() == Sex.*FEMALE*){  
 Arrays.*fill*(visited,false);  
 queue.clear();  
  
 visited[v] = true;  
 alternatingPath[v] = -1;  
 queue.push(v); // w tym miejscu uruchamiamy BFS do utworzenia ścieżki rozszerzającej  
  
 while (!queue.isEmpty()){  
 x = queue.getFirst();  
 xPeopleNode = (PeopleNode) this.getNode(x);  
 queue.removeFirst();  
  
 if(xPeopleNode.getPerson().getSex() == Sex.*FEMALE*){ line 1  
 //jezeli w kolejce trafiamy na kobiete, to w kolejce umieszczamy wszystkie nieodwiedzone sasiednie wierzcholki  
 for (Integer y: xPeopleNode.getNeighboursList()){  
 if(!visited[y]) {  
 visited[y] = true;  
 alternatingPath[y] = x;  
  
 queue.add(y);  
 }  
 }  
 }  
  
 else if(matching[x] > (-1)){ line 2

//tutaj obsługujemy już zajętego kawalera, a skojarzoną z nim pannę dodajemy do do koleki, w ten sposób tworzymy ścieżkę naprzemienną  
 alternatingPath[matching[x]] = x; line 4  
 visited[matching[x]] = true;  
 queue.add(matching[x]);  
 }  
 else{ line 3

// w tym kroku obsługujemy wierzchołek, który (jak wynika z poprzednich warunków), musi byc mężczyzną, który nie ma przyporządkowanej sobie panny  
 while(alternatingPath[x] > (-1)){

//poruszamy się po wyznaczonej ścieżce w celu znalezienia wolnej kobiety, przy czym zamieniamy krawędzie skojarzone I nieskojarzone  
 xPeopleNode = (PeopleNode) this.getNode(x);  
 if(xPeopleNode.getPerson().getSex() == Sex.*MALE*){

//tutaj zamieniamy krawędzie skojarzone i nieskojarzone  
 matching[x] = alternatingPath[x];  
 matching[alternatingPath[x]] = x;  
 }  
 x = alternatingPath[x];  
 }  
 break;  
 }  
 }  
 }  
 }  
  
 long endTime = System.*nanoTime*();  
 this.elapsedTime = endTime - startTime;  
  
 tableOfMatches = matching;

// zwracamy tablicę przyporządkowań jaką udało nam się stworzyć, na tym etapie nie jest sprawdzane czy wszyscy mają partnera. W tablicy może znaleźć się wartość -1, co oznacza, że poszukiwane przyporządkowanie nie istnieje.  
 return matching;  
}

1. Opis pojęć:

* Krawędź skojarzona – krawędź pomiędzy, mężczyzną a kobietą którzy zostali przyporządkowani do siebie
* Krawędź nieskojarzona(swobodna) – krawędź pomiędzy mężczyzną a kobietą, którzy NIE zostali do siebie przyporządkowani
* Ścieżka naprzemienna – ścieżka przebiegająca na przemian przez krawędzie swobodne i skojarzone. Zawsze rozpoczyna się od krawędzi swobodnej.

1. Opis zastosowanego algorytmu  
     
   W głównej pętli obsługujemy tylko kobiety, który nie zostały jeszcze przyporządkowane.   
     
   Jeżeli taką znajdziemy (line 1) to   
   tworzymy nową kolejkę, w której są umieszczeni wszyscy nieodwiedzeni mężczyźni z nią sąsiadujący.   
   Jednocześnie do tablicy pomocniczej alternatingPath dla każdego z mężczyzn jest przypisywany numer wierzchołka kobiety, jako numer wierzchołka nadrzędnego.  
     
   Dalej zajmujemy się kolejnym wierzchołkiem umieszczonym w kolejce.  
     
   Jeżeli ten wierzchołek to mężczyzna, który nie jest do nikogo przyporządkowany (line 3, wynika to z poprzednich warunków), to poruszamy się po ścieżce naprzemiennej w poszukiwaniu wolnej panny, do której możemy go przyporządkować. Poruszamy się ścieżką tak długo(po drodze zamieniając ze sobą skojarzone i nieskojarzone krawędzie, dopóki nie dotrzemy do wierzchołka, który nie ma swojego wierzchołka nadrzędnego.  
     
   Ostatni rozpatrywany przypadek podczas analizowania wierzchołków w kolejce to natrafienie na mężczyznę, który ma już przyporządkowaną sobie kobietę (line 2). W tym kroku zostaje tworzona ścieżka naprzemienna, bo umieszczamy w tablicy alternatingPath znalezionego mężczyznę, jako wierzchołek nadrzędny dla kobiety, do której został wcześniej   
   przypisany (line 4), jednocześnie wrzucamy tę kobietę do kolejki.  
     
   Po przeanalizowaniu, wszystkich wierzchołków w kolejce wracamy do obiegu głównej pętli.
2. Podsumowanie  
     
   Algorytm w głównej pętli odwiedza wszystkie nieodwiedzone wcześniej kobiety, a w pętli od razu umieszcza na kolejce wszystkich nieodwiedzonych wcześniej sąsiadów i-tej kobiety. Zatem jeżeli nie odwiedzimy, któregoś z wierzchołków w trakcie tych dwóch czynności to znaczy, że został on już odwiedzony np. Podczas wykonywania akcji w samej kolejce.

Bartosz Cybulski

Nr indeksu 283 721.