Enkele antwoorden van opgaven bij werkcollege 1 Algoritmiek

7.
```
void telvoorEuler ( buur* graaf[n] ) {
  int totaal = 0;
  for ( int i=0; i<n; i++ ) {
    teller = 0;
    buur* hulp = graaf[i];
    while ( hulp != NULL ) {  // buurlijst aflopen
      teller ++;
      hulp = hulp->volgende;
    }
    if ( teller%2 == 1 )  // oneven
      totaal ++;
  }  // for
  if ( totaal <= 2 )
    cout << "hooguit twee..." << endl;
  else
    cout << "meer dan twee..." << endl;
}
```

8. (a)
```
int takken ( buur* graaf[n] ) {
  int count = 0;
  for ( int i=0; i<n; i++ ) {
    buur* hulp = graaf[i];
    while ( hulp != NULL ) {  // buurlijst aflopen
      count++;
      hulp = hulp->volgende;
    }
  }
    // alles 'per ongeluk' dubbel geteld -> corrigeren
  return count/2;
}
```

(b)
```
int takken2 ( int graaf[n][n] ) {
  int count = 0;
  for ( int i=0; i<n; i++ ) {
    for ( int j=0; j<n; j++ ) {
      if ( graaf[i][j] > 0 ) {
        count++;
      }
    }
  }
    // alles 'per ongeluk' dubbel geteld -> corrigeren
  return count/2;
}
```

Variant: alleen deel rechtsboven de diagonaal van de adjacency matrix aflopen, of juist alleen het deel linksonder de diagonaal. Dan hoef je na afloop niet door 2 te delen.

(c) De adjacency list representatie is het meest efficiënt, omdat je daarbij alleen maar kijkt naar de bestaande takken. Bij de adjacency matrix representatie loop je ook niet-bestaande takken af.

9. (a) Omkeren van een pijl (i, j) met adjacency matrix `int graaf[n][n]` .
Er loopt tevoren een tak (i, j), maar geen tak (j, i), dus `graaf[i][j] = 1` en `graaf[j][i] = 0`. Dit draaien we om.

```
void draaiom ( int graaf[n][n], int i, int j ) {
  graaf[i][j] = 0;
  graaf[j][i] = 1;
}
```

(b) Omkeren van een pijl (i, j) met adjacency list `buur* graaf[n]` .
De lijsten zijn niet per se gesorteerd. Er loopt tevoren een tak (i, j), maar geen tak (j, i), dus j komt buur in lijst `graaf[i]` en i komt niet voor in lijst `graaf[j]` .

Idee voor algoritme:
- j in buurlijst van i opzoeken
- j uit buurlijst van i verwijderen
- i vooraan in buurlijst van j plaatsen.

```
void draaiom2 ( buur* graaf[n], int i, int j ) {
  buur* hulp = graaf[i];
  buur* vorige = NULL;
  while ( hulp->knoopnummer != j ) { // knoop j zoeken
    vorige = hulp;
    hulp = hulp->volgende;
  }
    // haal buur j uit lijst
  if ( vorige == NULL ) { // j is eerste buur van i
    graaf[i] = hulp->volgende;
  }
  else { // j is niet eerste buur van i
    vorige->volgende = hulp->volgende;
  }
  delete hulp;  // gooi buur j weg

    // voeg nu i vooraan in de buurlijst van j toe
  hulp = new buur;
  hulp->knoopnummer = i;
  hulp->volgende = graaf[j];
  graaf[j] = hulp;
}
```