**Ülesanne 3**

Kuna loend on juba osaliselt sorteeritud siis on algoritmil vähem tööd teha. Iga elemendi sisestamisel peab liikuma läbi ainult juba osaliselt sorteeritud osast ning vahetavad kohti ainult nendega mis on suuremad kui lisatavad elemendid. Kui loend oleks täielikult sorteerimata siis peaks algoritm läbi käima kogu loendi ja vahetama järjekorda mitmeid kordi rohkem. Seetõttu on Insertion Sort selliste osaliselt sorteeritud loendite puhul ka tõhusam.

**Ülesanne 4**

1. Kõikide eelmainitute algoritmide ajakulu suureneb ruutsõltuvusest loendi suurusega ehk kõikidel neil on ajakompleksus.
2. Counting Sort. Selle ajakulu suureneb lineaarselt loendi suurusega. Kuna võimalikud väärtused on piiratud siis see muudab sellises olukorras just selle kõige efektiivsemaks.

**Ülesanne 5**

1. Sortimisalgoritm on stabiilne kui see säilitab sama elemendijärjestuse kui me võrdleme elemente, mille väärtused on võrdsed ehk kui need on võrdsed ja igakord täpselt samas järjekorras siis on algoritm ka stabiilne.

Näiteks: Armand (30), Rivo (23), Tanel (30), Marko (44) ja kui me sorteerime selle vanuse järgi siis stabiilne algoritm tagab meile et tulemus on alati sama ehk Rivo(23), Armand(30), Tanel(30), Marko(44).

1. See suudab kohandada oma käitumist vastavalt sisendandmetele. Algoritm suudab ära kasutada loendi juba olemasolevat korraldust ja sellega seotud informatsiooni, et kiiremini tulemus saavutada. Eelmainitust on adaptiivne Insertion Sort. See kasutas juba osaliselt sorteeritud järjekorda.

**Boonus**

1. Bubble Sort: Esimese läbimise tulemus: [3, 5, 4, 7, 6, 2, 8]
2. Selecction Sort: Esimese läbimise tulemus: [2, 3, 5, 4, 7, 6, 8]
3. Insertion Sort: Esimese läbimise tulemus: [3, 8, 5, 4, 7, 6, 2]