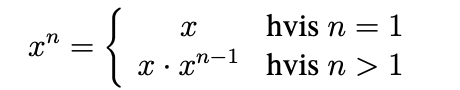
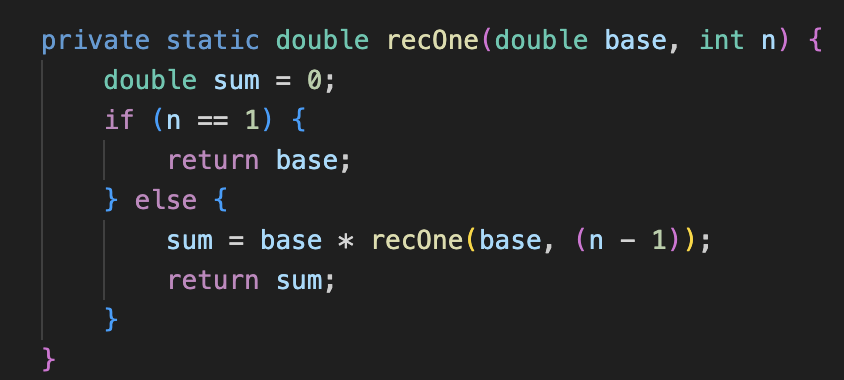
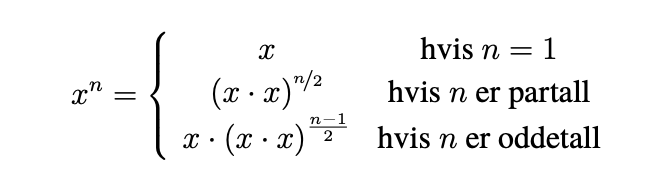
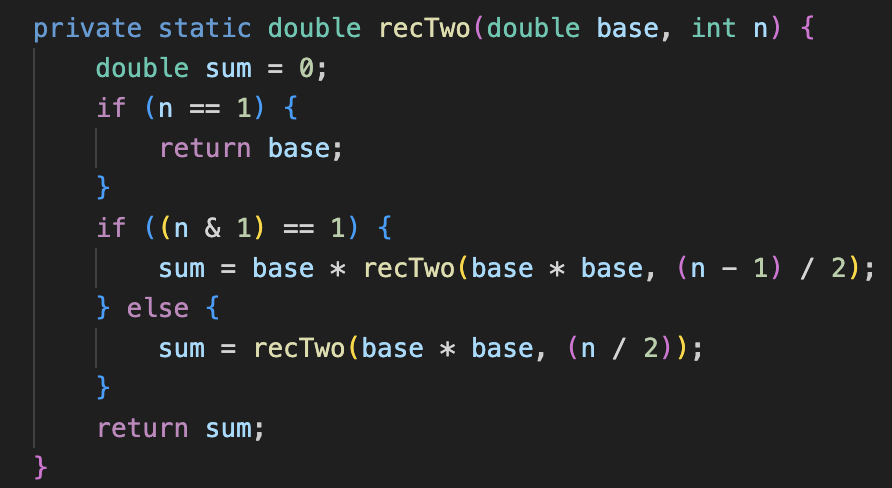
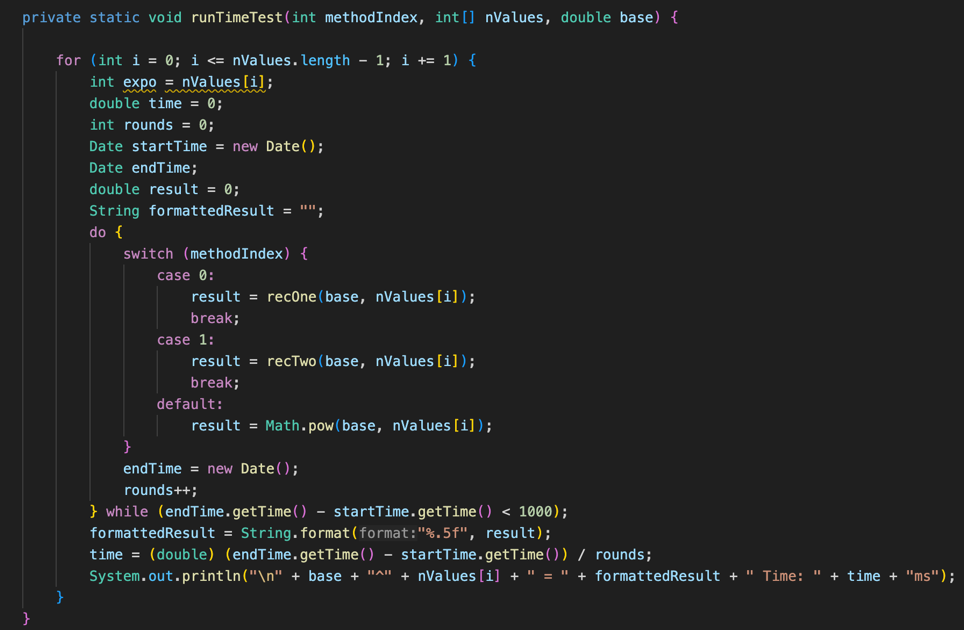
AlgDat øving 2

**Metode 1 –**

Metoden recOne() er rekursiv og beskrives matematisk ved bildet til høyre. Metoden har ingen løkker og kjører gjennom dataen n-antall ganger. Metoden har derfor en tidskompleksitet på: . Som da blir O(n) (lineær).

**Metode 2 –**

Metoden recTwo() har to forskjellige metoder den bytter mellom men til hver iterasjon brukes kun en av disse. Tidskompleksiteten kan gis ved:

****Der *a* er antall rekursive kall i metoden (1), *b* er brøkdelen av datasett vi behandler i et rekursivt kall (2) og *k* beksriver hvordan arbeidsmengden vokser iforhold til inputen/datasett (for-løkker) denne blir 0 så da gir formelen tidskompleksitet:

**Metode 3 og Tidtaking –**

Metoden runTimeTest() tar for seg tidtakingen for en av rekursjonsmetodene, med gitte n-verdier og en base. Det er denne metoden som tar og printer tid. Metode 3 brukes i switch-setningen ved default-caser.

**Analyse av tidsmålinger -**

Utskriften for main-metoden vises nedenfor og starter med å vise at de rekursive metodene og Math.pow-metoden får det samme riktige resultat. Math.pow og den lineære rekursive metoden kjører gjennom nValues, mens nValues2 kjører gjennom flere og høyere *n* for å få bedre forståelse for utvikling av tid.

Man ser at alle metodene gir samme svar for 5^11. Videre er det utskrift for de tre metodene med grunntall 1.002 og stigende eksponenter.

Man ser også at for hver dobling av n hos den lineære metoden så dobles tiden omtrentlig.

Ved den logaritmiske metoden ser man at den stiger relativt raskt (*n* = 2001 er relativt lavt, og det kan oppstå ujevnheter) før den flater ut mer og mer ved høyere *n*. Dette er gjenkjennbart med logaritmiske funksjoner. Math.pow() ligger derimot relativt likt på alle verdier av *n*, på rundt *.*

