ChatGPT

Verslag: Implementatie van Iteratieve Merge Sort in C++

Inleiding

In dit verslag wordt de aanpak besproken voor het implementeren van een iteratieve versie van het Merge Sort algoritme in de programmeertaal C++. Deze implementatie is gebaseerd op de Python-code uit opdracht 2.1, en vormt de basis voor de latere uitbreiding naar een multithreaded versie in opdracht 2b.

Aanpak

De keuze is gemaakt om een iteratieve in-place versie van Merge Sort te implementeren. Dit betekent dat er geen gebruik wordt gemaakt van recursie of van extra arrays tijdens het sorteerproces. De implementatie volgt nauwgezet het patroon zoals beschreven in de Python-variant:

1. **Eenheidsgrootte (unit)**: De lijst wordt in steeds grotere blokken samengevoegd: eerst per 2 elementen, dan per 4, vervolgens per 8, enzovoort.
2. **In-place merging**: Binnen elk blok worden elementen samengevoegd zonder het gebruik van extra geheugen. Wanneer een element rechts van het midden kleiner is dan een element links, worden de elementen verschoven om de volgorde te behouden.
3. **Stabiliteit**: De sortering blijft stabiel doordat bij gelijke elementen de volgorde behouden blijft (door gebruik van `<=`).

De implementatie werd geschreven in standaard C++ met gebruik van de STL `vector` klasse.

Code

De volledige code is te vinden in de volgende sectie:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

void merge_sort(std::vector<int>& xs) {
    int n = xs.size();
    int unit = 1;

    while (unit <= n) {
        for (int h = 0; h < n; h += unit * 2) {
            int l = h;
            int r = std::min(n, h + 2 * unit);
```

```
            int mid = h + unit;
            if (mid >= r) continue;

            int p = l;
            int q = mid;

            while (p < mid && q < r) {
                if (xs[p] <= xs[q]) {
                    p++;
                } else {
                    int temp = xs[q];
                    for (int k = q; k > p; --k) {
                        xs[k] = xs[k - 1];
                    }
                    xs[p] = temp;
                    p++;
                    mid++;
                    q++;
                }
            }
        }
        unit *= 2;
    }
}

int main() {
    std::vector<int> xs = {5, 2, 4, 6, 1, 3, 2, 6};
    merge_sort(xs);

    for (int x : xs) {
        std::cout << x << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

Complexiteitsanalyse

Tijdcomplexiteit

- In elke fase wordt de volledige array van lengte n doorlopen en samengevoegd.
- Het aantal fasen is $\log_2(n)$.
- Totale tijd: $O(n \log n)$.

Ruimtecomplexiteit

- De implementatie is volledig in-place; er wordt geen extra geheugen gebruikt buiten de input-array.

- Ruimtecomplexiteit: $O(1)$.

Conclusie

De iteratieve in-place Merge Sort is succesvol omgezet van Python naar C++. De keuze voor deze aanpak levert een stabiele en geheugen-efficiënte oplossing op die bovendien geschikt is voor uitbreiding naar een multithreaded variant. Deze implementatie vormt een goede basis voor de verdere uitwerking in opdracht 2b.