A ideia do ShellSort é ser um insertion sort melhorado. O que o insertion sort fazia é, dado um elemento na posição i, ele ia jogando este elemento o mais para traz possível, porém ele sempre ia jogando para traz de 1 em 1, o que não é muito bom.

Com isso, o ShellSort melhora esta ideia em que, invés de jogar o elemento para traz de 1 em 1 ele joga para traz de Gap em Gap.

Logo, dado um Gap, ele vai começar a analisar o primeiro elemento q está exatamente no Gap.

* Vai jogar para traz quanto maior possível, sempre comparando v[i - gap] com v[i]
* Dps de fazer cada comparação esta comparação e vc fez um swap, vc vai fazer uma outra comparação, dessa vez de v[i - 2.gap] com v[i - gap], até que vc n possa fazer mais comparações ou que v[i - gap] seja menor do que v[i]
* Dps disso vc incrementa i
* Dps de fazer para todos os I vc diminuiu o gap por 2
* Vai fazer isso enquanto o gap for maior que 0

void ShellSort(int v[], int n, int h) {

int gap = h;

while (gap > 0) {

for (int i = gap; i < n; i++){

int j = i;

while(v[j - gap] > v[j]){

swap(v, j - gap, j);

j -= gap;

}

}

gap /=2 ;

}

}

A complexidade no caso médio é O(n.log²n) - não sei provar.

No pior caso é O(n²) - também não sei provar

