1

R3: $<, >, <=, >=$
ordre lexicographie sur les pair est deja dans la stl

1.1 Multimaps

(Requires a less-than comparison function.)

$O(3n_T)$: jede Kante jedes Dreiecks als Schlssel
$|K|$ $O(log(n_T))$: Zugriff auf Container fur jede Kante
$\Rightarrow O(n_T log(n_T))$
speichern in 2dim Array: Zugriff in konstanter Zeit

unordered map(hash tables) has constant time performance on all operations provided no collisions occur. When collisions occur, traversal of a linked list containing all elements of the same bucket (those that hash to the same value) is necessary, and in the worst case, there is only one bucket; hence O(n)

1.2 Listen

$O(3n_T)$, push front(): jede Kante jedes Dreiecks einfgen
$O(NlogN)$, container size N: sort(): definiere hierfr $<$ fr R3 (Nach ersten und dann nach zweitem Element sortieren etc)

doppelt verlinkt! und dann mit Kante und Sommet prev und next von Listenelement!
konstanter Zugriff mit Iterator!

$O(3n_T)$: pop front und speichern in 2 dim array

Alternative fr 2D array fr konstanten Zugriff?
Daten knnten direkt in 2D array gespeichert werden, Listen und Maps berflssig

1.3 aire

orientierter Flcheninhalt

$$(a_i, b_i, p) = (a_i \times b_i) \cdot p = \det(a_i, b_i, p)$$

bei det Vektoren in Zeilen