*Obiettivo*

ottimizzare la memorizzazione di informazioni in una struttura dati che sia efficiente soprattutto in termini di memoria (oltre che ricerca e rimozione)

Inizialmente non si nulla da salvare in memoria, ma si deve valutare l'immagine così come è, quindi calcolando tutto durante l'analisi dell'immagine stessa.

Realizzare l’algoritmo che sia in grado di funzionare sia in 2D che 3D. Se si riesce, farlo per qualsiasi dimensione, in generale.

*Pseudocodice*

Immagine che contiene testo

Descrizione generata automaticamente

*Esempio Jiri*

Immagine che contiene testo, lavagnabianca

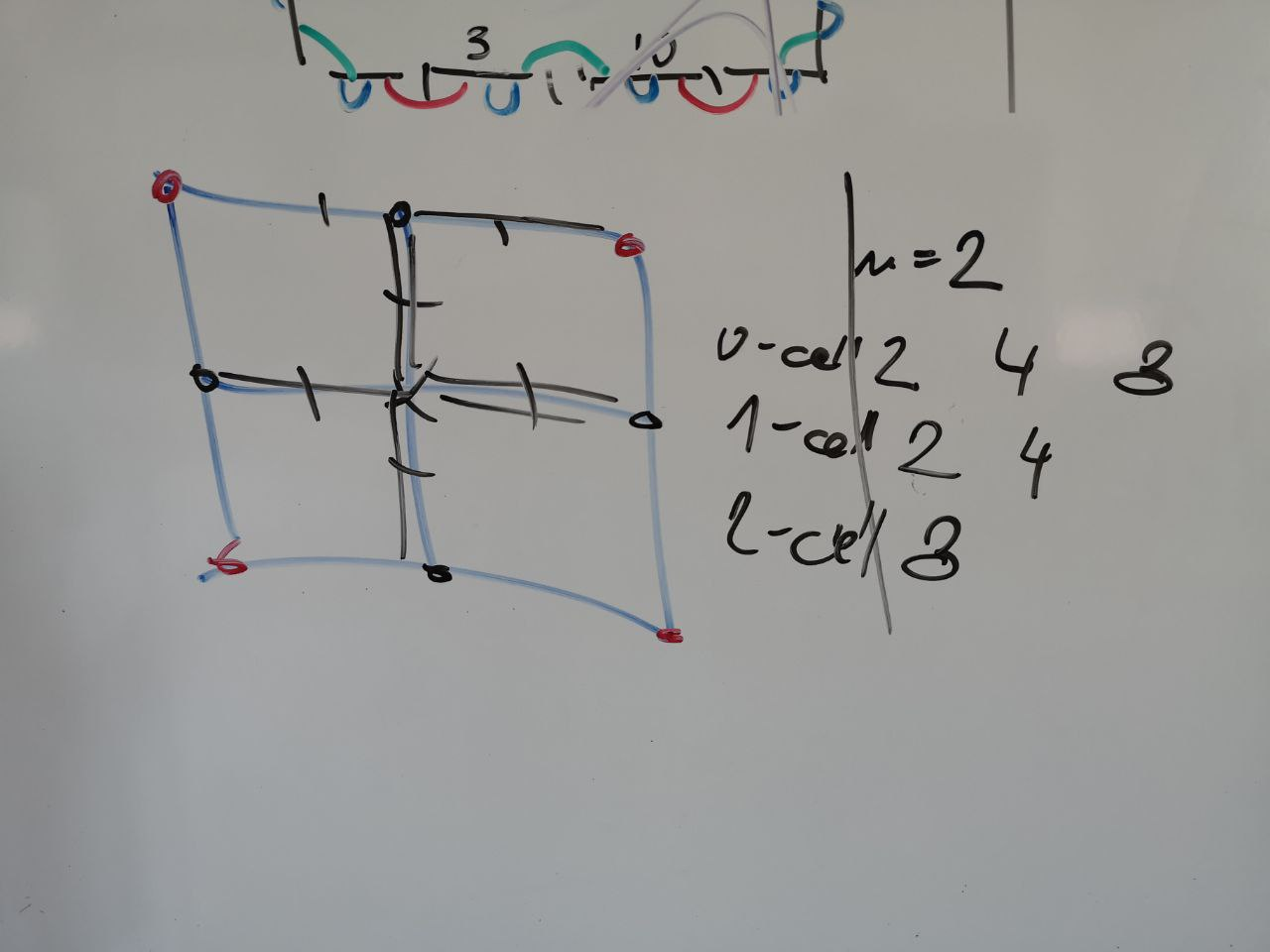
Descrizione generata automaticamente

Bisogna mantenere in memoria solo le informazioni che riguardano il *dart* che è stato rimosso.

*Operazione*

L’operazione principale da tenere in considerazione è quella di *rimozione* (o *contrazione* che ne è il duale).

*Esempio Jiri*



Consideriamo un’immagine rappresentata da una 2-Gmap.

Possiamo distinguere due tipi di vertici:

* CORNER (indicati con il colore rosso)
* E quelli indicati con il colore nero

Come deduciamo: n=2.

* 0-cell -> rappresenta i vertici
  + Distinguiamo 3 casi:
    - Vertice all’angolo: #dart = 2 (prendendo come riferimento quello in alto a destra, si ha un dart che va verso sinistra e un dart che va verso il basso)
    - Vertice al centro dell’immagine: #dart = 8 (2 per ogni lato dati dallo split degli edge)
    - Vertice tra i due all’angolo che divide l’edge in due: #dart = 4
* 1-cell -> rappresenta gli edge
  + Distinguiamo 2 casi:
    - (ovale rosso): #dart = 4 (split degli edge da cui poi si ricavano 4 dart, 2 ognuno)
    - (ovale giallo): #dart = 2 (edge limite, fa parte del *bound* e quindi non può essere scomposto, di conseguenza si hanno solo 2 dart)
* 2-cell -> rappresenta le facce
  + Distinguiamo un unico caso:
    - Considerando ogni faccia, questa è sempre caratterizzata da un edge per lato. Decomponendo gli edge per ottenere i dart, se ne ottengono: #dart = 8.

## After reading nth-paper and clarifications

Il paper letto è [resources\FB\_implicit encoding of contracted nGmaps.pdf](resources/FB_implicit%20encoding%20of%20contracted%20nGmaps.pdf)

Il primo step ha visto il ragazzo utilizzare una codifica implicita della griglia di pixel da cui si è ricavato, per ognuno, le coordinate. Inoltre, ha deciso di utilizzare una codifica posizionale per i dart in una certa base *b*.

Per quanto riguarda le involuzioni, queste sono memorizzate in una *lookup-table*.

Il secondo step lo ha visto rimuovere la 1-cella per unire due pixel dove però ha evidenziato l’esistenza di una problema.

Il terzo step ha previsto la semplificazione delle membrane, ovvero (se ho capito bene), la rimozione delle 0-celle.

L’ultimo step prevede la ricostruzione della nGmap dopo aver effettuato queste operazioni di rimozione e contrazione.

La complessità di questo algoritmo dipende dal tipo di struttura dati che si utilizza. Una prima soluzione potrebbe essere quella di utilizzare CGAL, dove al posto dell’insieme di dart si può ragionare utilizzando una *hash-map o dizionario* che deve mantenere un’associazione bilaterale tra i dart in gioco.

* Obiettivo: trovare una struttura dati che consenta di mantenere pochi dati, che sia bilaterale e veloce.