ELABORAZIONE DELLE IMMAGINI

SEGMENTAZIONE DELLE IMMAGINI: REGION GROWING – SPLIT AND MERGE

SEGMENTAZIONE BASATA SULLE REGIONI

- Denotiamo con ***R*** la regione occupata dall'**immagine**
- La **segmentazione** dell'immagine consiste nel **partizionare** *R* in *n* **sottoregioni** $R_1, R_2, \dots, R_n$ tali che
  - $\bigcup_{i=1}^{n} R_i = R$ (segmentazione completa)
  - R_i è un insieme connesso $i = 1, 2, \dots, n$ (pixel 4/8 connessi)
  - $R_i \cap R_j = \emptyset$ (regioni disgiunte)
  - $Q(R_i)$ = Vero per $i = 1, 2, \dots, n$ (tutti i pixel in R_i rispettano la proprietà in Q)
  - $Q(R_i \cup \mathrm{R_j})$ = Falso $\forall\, R_i\ e\ R_j$ adiacenti (pixel in regioni adiacenti devono avere proprietà diverse)
- Q è un predicanto logico definito sui punti di una regione R_i

REGION GROWING

- La tecninca del **region growing** (accrescimento di regione) consiste nel **raggruppare** i **pixel** in **regioni** più **grandi** in base a **criteri** predefiniti
- Il procedimento **inzia** da punti iniziali detti **seed** e si **propaga** ai pixel **adiacenti** che rispettano delle **proprietà** predefinite, i quali vengono **aggiunti** alla regione
- Se l'informazione relativa ai **seed non** è **disponibile**, per **ogni pixel** si calcolano delle **proprietà** che vengono usate per l'**assegnazione** dei **pixel** alle **regioni**

REGION GROWING

- Il criterio di **similarità** dipende dal tipo di immagine
- Es.: per **immagini** in scala di **grigio** si usano descrittori basati sui **livelli** di **intensità**
- I descrittori devono sempre essere associati alle **proprietà** di **connettività**
- Ragruppare **pixel** "simili" ma **non adiacenti** può generare **segmentazioni inconsistenti**

REGION GROWING

- Un altro problema riguarda la **regola d'arresto**, utile per arrestare l'**accrescimento** della regione quando i **pixel non soddisfano** più i **criteri** di inserimento
- Considerare solo i **descrittori locali** (es. livello di intensità) non è sufficiente perchè non si tiene conto della "**storia**" del processo di accrescimento
- È opportuno **conisderare** le caratteristiche di tutti i **pixel** che già sono stati **inseriti** nella regione

REGION GROWING

- Sia ***f(x,y)*** l'immagine di input
- Sia ***S(x,y)*** la matrice dei **seed** che assegna il valore 1 alle **posizioni dei seed** e 0 alle altre posizioni
- Sia ***Q*** un **predicato** da aplicare ad ogni pixel
- I passi dell'**algoritmo** sono i seguenti

1. Formare l'immagine f_Q che nel punto (x,y) contiene il valore 1 se $Q(f(x,y))$ è vero altrimenti contiene il valore 0
2. Aggiungere ad ogni seed i pixel impostati ad 1 in f_Q che risultano [4,8]-connessi al seed stesso
3. Marcare ogni componente connessa con un'etichetta diversa

REGION GROWING

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

REGION GROWING

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

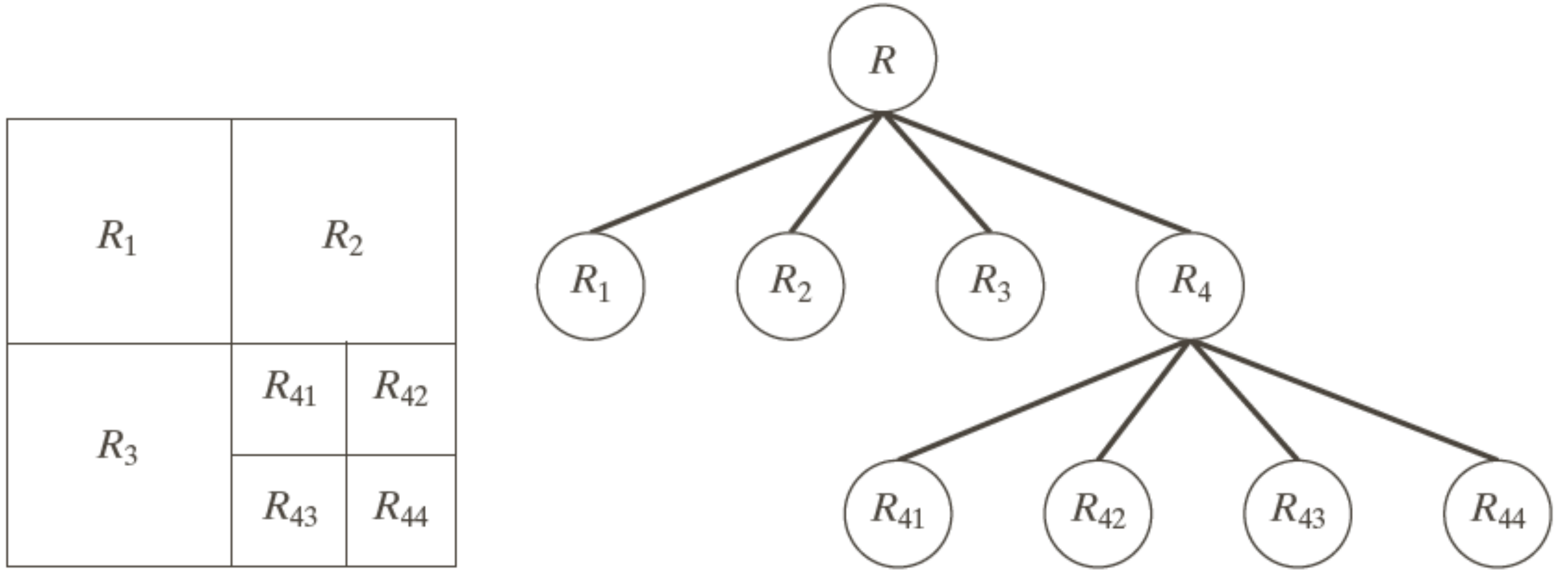
1	1	9	9	9
1	1	9	9	9
5	1	1	9	9
5	5	5	3	9
3	3	3	3	3

REGION GROWING

SPLIT AND MERGE

- Un approccio alternativo consiste nel **dividere** (**split**) l'immagine in **regioni disgiunte** di forma e dimensioni arbitrarie e successivamente **fonderle** (**merge**) in base a dei **criteri di similarità**
- Sia **R** la **regione** corrispondente all'intera **immagine** e **Q** un **predicato**
- È possibile **dividere R** in **regioni** sempre più **piccole** finchè il **predicato** non risulta **vero**
- Una strategia molto utilizzata consiste nel **dividere** le **regioni** in **quadranti** (mediante **quadtrees**, alberi di quadranti)
- Partendo da R, se **Q(R)=falso** si **divide** R in **quattro quadranti**
- I **quadranti** per cui **Q** è **falso** si **dividono** ulteriormente **in quadranti** e così via **finchè** il **predicato non risulta vero**

SPLIT AND MERGE



SPLIT AND MERGE

- Al **termine** della fase di **splitting**, la **partizione** finale potrebbe contenere **regioni adiacenti** con caratteristiche **simili**
- Per questo motivo queste **regioni** possono essere **fuse** (lo schema illustrato prevede che $Q(R_i \cup \mathrm{R_j})$ = Falso $\forall\, R_i\ e\ R_j$ adiacenti)
- Si considerino due **regioni adiacenti** $\boldsymbol{R_i\ e\ R_j}$, se $\boldsymbol{Q(R_i \cup R_j)}$ **=Vero** allora è possibile effettuare il merge delle due regioni

SPLIT AND MERGE

- I passi dell'algoritmo sono i seguenti

1. **Dividere** in quattro quadranti tutte le regioni per cui il predicato Q risulta falso
2. Quando non è più possibile dividere le regioni, applicare il processo di **merging** a tutte le regioni adiacenti R_i e R_j, per cui $Q(R_i \cup \mathrm{R_j})$ =Vero
3. Il processo **termina** quando **non** è **più** possibile effettuare **unioni**

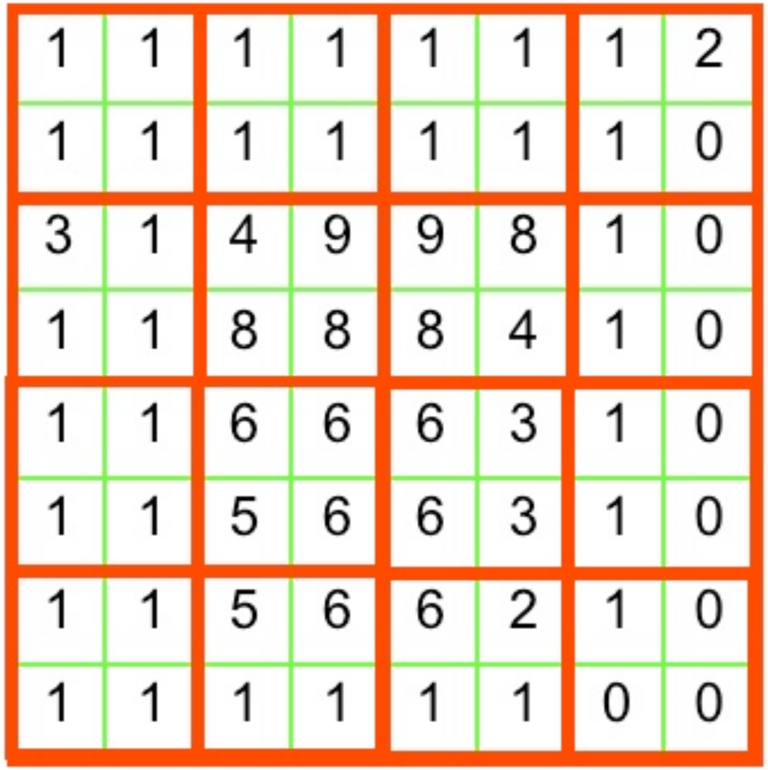
- Solitamente si definisce una **dimesione minima** della **regione** oltre la quale non si effettua lo split
- Per ragioni di efficienza, la fase di merge si può eseguire se il predicato è vero per le singole regioni adiacenti (non si effettua l'unione)

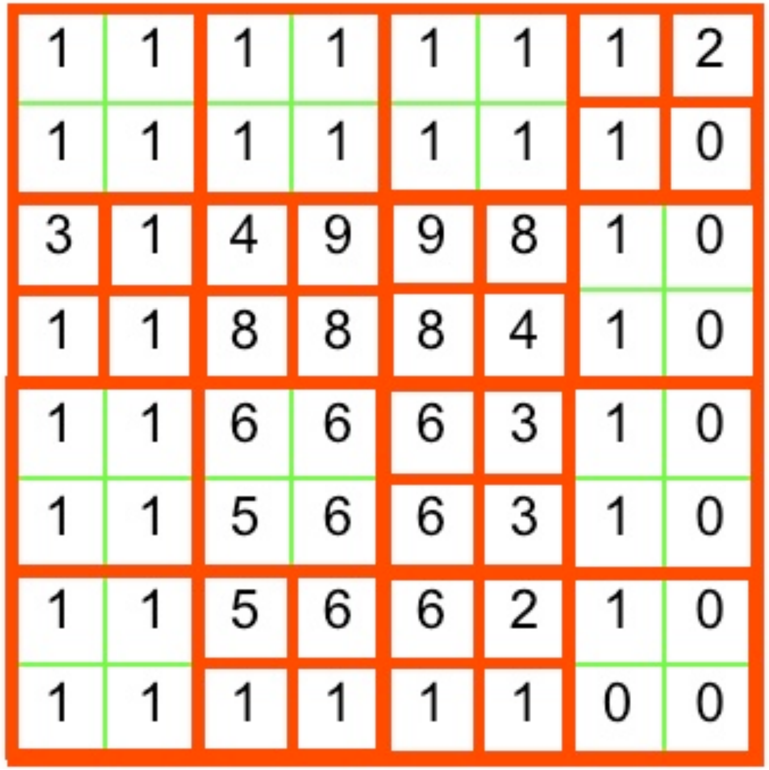
SPLIT AND MERGE

1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

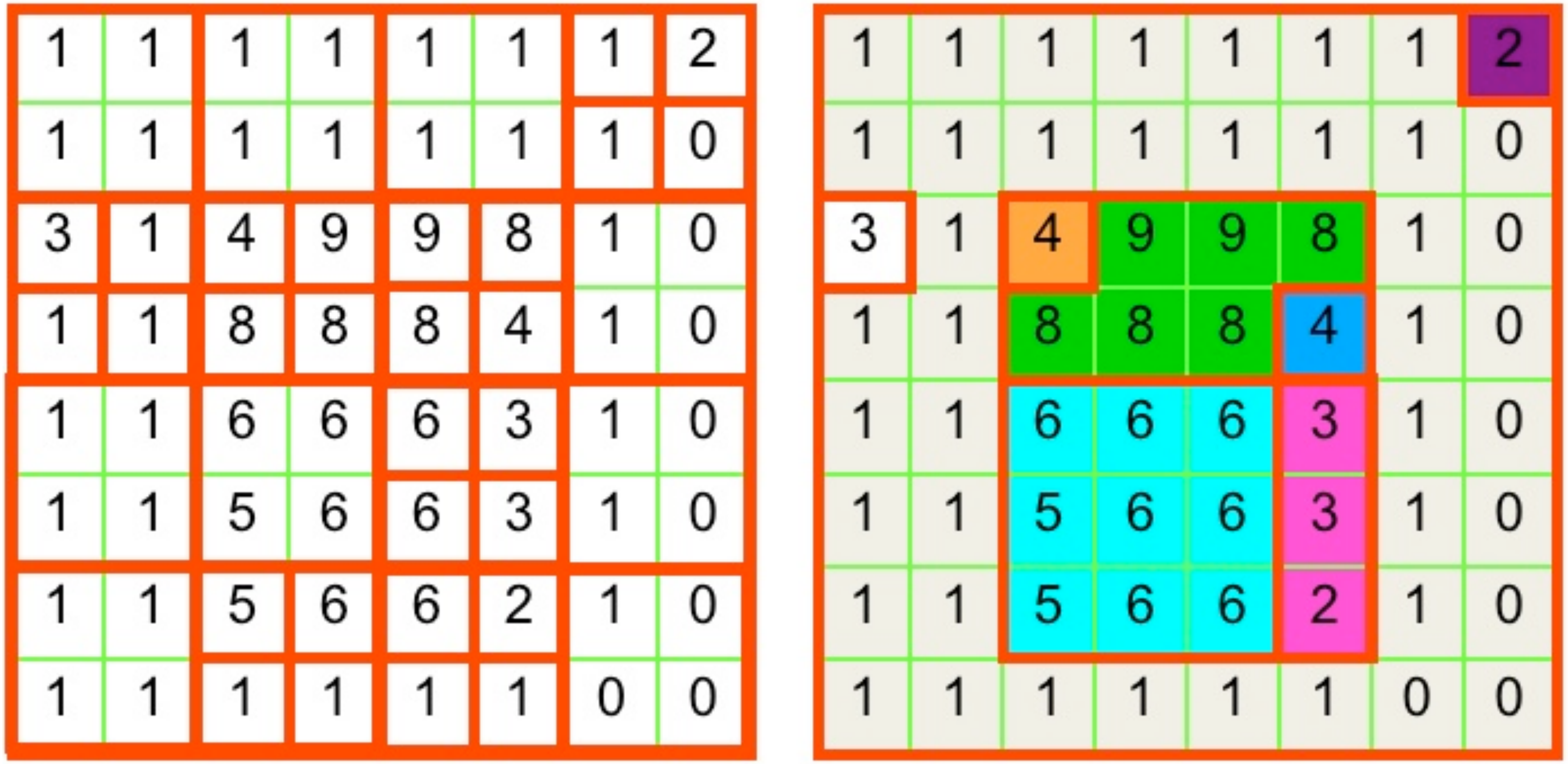
1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0
1	1	8	8	8	4	1	0
1	1	6	6	6	3	1	0
1	1	5	6	6	3	1	0
1	1	5	6	6	2	1	0
1	1	1	1	1	1	0	0

SPLIT AND MERGE

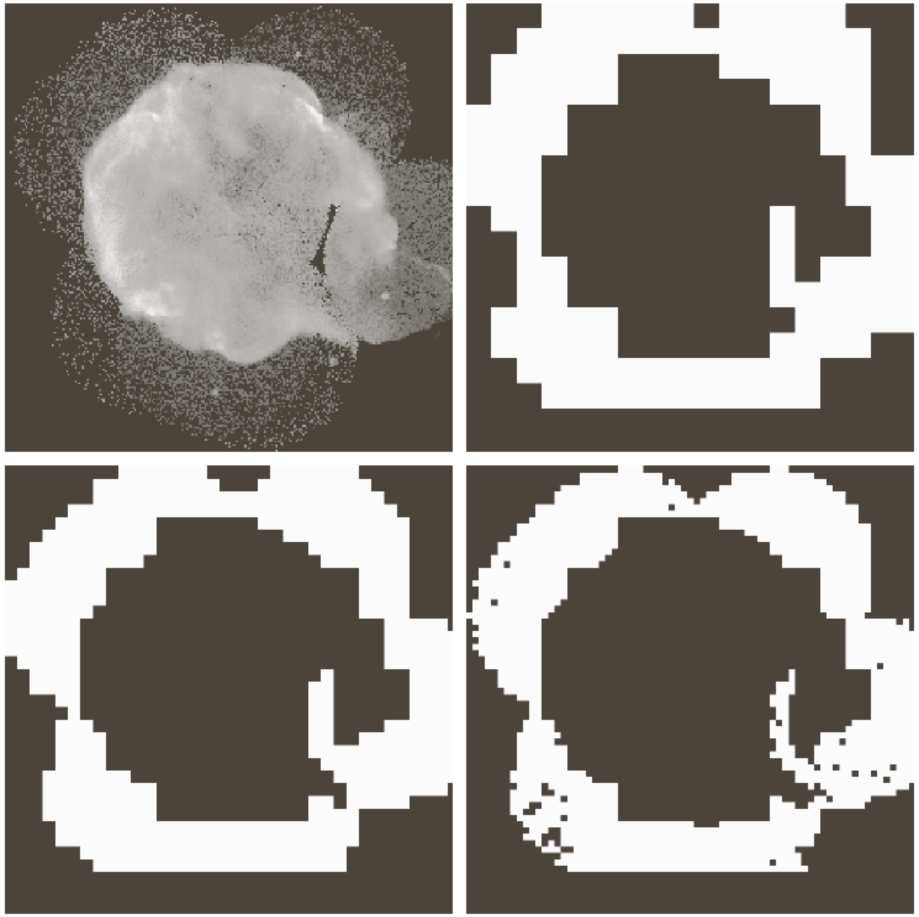




SPLIT AND MERGE



SPLIT AND MERGE



a	b
c	d

$$Q := \sigma > \alpha \text{ AND } 0 < m < \beta$$

- (b) 32×32
- (c) 16×16
- (d) 8×8

ESERCIZI

- Provare il codice region growing
- Implementare l'algoritmo split and merge
- Un algoritmo per trovare i nodi adiacenti di una dato nodo (es. nord) [Computational Geometry – Algorithms and Applications, deBerg et al.]

Algorithm NORTHNEIGHBOR($\nu, \mathcal{T}$)
Input. A node ν in a quadtree $\mathcal{T}$.
Output. The deepest node ν' whose depth is at most the depth of ν such that $\sigma(\nu')$ is a north-neighbor of $\sigma(\nu)$, and **nil** if there is no such node.

1. **if** $\nu = root(\mathcal{T})$ **then return nil**
2. **if** $\nu =$ SW-child of $parent(\nu)$ **then return** NW-child of $parent(\nu)$
3. **if** $\nu =$ SE-child of $parent(\nu)$ **then return** NE-child of $parent(\nu)$
4. $\mu \leftarrow$ NORTHNEIGHBOR($parent(\nu), \mathcal{T}$)
5. **if** $\mu =$ **nil or** μ is a leaf
6.   **then return** μ
7.   **else if** $\nu =$ NW-child of $parent(\nu)$
8.   **then return** SW-child of μ
9.   **else return** SE-child of μ