ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A new algorithmic framework for basic problems on binary images

T. Asano [a,*], L. Buzer [b,c], S. Bereg [d]

[a] JAIST, Japan
[b] Université Paris-Est, LABINFO-IGM, UMR CNRS 8049, France
[c] Department of Computer Science, ESIEE, France
[d] Department of Computer Science, University of Texas at Dallas, USA

ARTICLE INFO

Article history:
Received 26 September 2014
Received in revised form 14 February 2016
Accepted 29 February 2016
Available online xxxx

Keywords:
Algorithm
Binary image
Computational complexity
Connected component
Connected components labeling
Connectivity
Constant work space algorithm
Grid graph
Small work space algorithm

ABSTRACT

This paper presents a new algorithmic framework for some basic problems on binary images. Algorithms for binary images such as one of extracting a connected component containing a query pixel and that of connected components labeling play basic roles in image processing. Those algorithms usually use linear work space for efficient implementation. In this paper we propose algorithms for several basic problems on binary images which are efficient in time and space, using space-efficient algorithms for grid graphs. More exactly, some of them run in $O(n \log n)$ time using $O(1)$ work space and the others run in $O(n)$ or $O(n \log n)$ time using $O(\sqrt{n})$ work space for a binary image of n pixels stored in a read-only array.

1. Introduction

The demand for embedded software is growing fast with the increased use of intelligent hardware such as scanners, digital cameras, and with the availability of other smart and portable equipments. One of the most important aspects and also a difference between ordinary software in computers and embedded software comes from the constraints on the size of local memory. For example, to design an intelligent scanner, a number of algorithms should be embedded in the scanner. In most of the cases, the size of the pictures is increasing while the amount of work space available for such software is severely limited. Thus, algorithms which require a restricted amount of work space and run reasonably fast are desired.

In this paper we propose several space-efficient algorithms designed for some fundamental image processing tasks, which essentially look for connected components of a grid graph. In one case we are requested to find the number of connected components or the largest component in the grid graph. All of the tasks that we consider have straightforward solutions if sufficient memory (typically, of size proportional to the size of the image) is available (see, for example, [13,17]). Solving the same tasks with restricted memory, without severely compromising the running time, is more of a challenge.

* Corresponding author.
E-mail addresses: t-asano@jaist.ac.jp (T. Asano), l.buzer@esiee.fr (L. Buzer), besp@utdallas.edu (S. Bereg).

http://dx.doi.org/10.1016/j.dam.2016.02.025
0166-218X/

Please cite this article in press as: T. Asano, et al., A new algorithmic framework for basic problems on binary images, Discrete Applied Mathematics (2016), http://dx.doi.org/10.1016/j.dam.2016.02.025

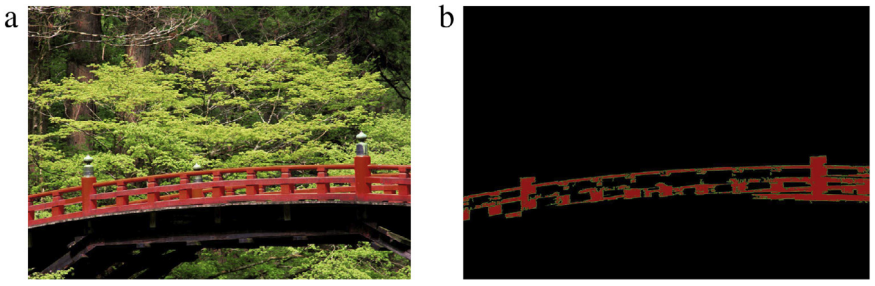

Fig. 1. Input color picture (a) and its largest component in a binary image defined by the same predicate f on color values(b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

1.1. Computational model with limited work space

We measure the space efficiency of an algorithm by the amount of work space used. Such space typically takes the form of pointers and counters (whose number of bits is at most the logarithm of the image size). Our objective is to formulate efficient algorithms that use only $O(1)$ or $O(\sqrt{n})$ such pointers and counters.

Throughout the paper we assume that an input binary image consists of $O(\sqrt{n})$ rows and $O(\sqrt{n})$ columns. We also assume a read-only array keeping an input binary image with random access to pixels.

1.2. Why read-only array?

Why do we assume a read-only array for an input binary image? Algorithms using read-only access allow parallel processing on the same image data. Another reason is to exclude possibility of a sophisticated mechanism to embed some information in an input array. Such a technique is used in implicit and succinct data structures (see e.g. [15]).

It is known that predicates can be used to transform images and save the memory. For example, suppose that we are interested in extracting an object region in a given color image $\mathcal{C}$ of n pixels and we know some color information characterizing the object. Then, using this information we can define a predicate f which determines in constant time whether a pixel value (color) is for the object. The pair $(\mathcal{C}, f)$ defines a binary image. In our setting, we can expect that the largest connected component in the binary image corresponds to the object region. To save memory space, we do not create a binary image from the input image since any pixel value can be computed in constant time. This is equivalent to assuming that the binary image is stored in a read-only array.

Fig. 1 shows such an example. A target object in Fig. 1(a) is a bridge which is characterized by color intervals, say Red [80, 255], Green [0, 80], and Blue [0, 255]. Fig. 1(b) shows the largest component.

1.3. Basic tasks and known results

Five basic tasks for a binary image considered in this paper are:

CCC—Connected Components Counting: Count the number of connected components in the image.

MERR—Minimum Enclosing Rectangles Reporting: Report the minimum enclosing (axis-parallel) rectangle of every connected component.

LCCR—Largest Connected Component Reporting: Report the locations of all the pixels of a largest connected component. Order of those pixels is arbitrary.

BILCC—Binary Image defined by Largest Connected Component: Output the binary image defined by the largest connected component in a given binary image in raster order. More exactly, output the binary image in which a pixel is white if and only if it is originally white and also belongs to the largest connected component in the given image.

CCL—Connected Components Labeling: Assign integral labels to all pixels so that any black pixel has label 0 and any two white pixels have the same positive integral label if and only if they belong to the same connected component.

LCCR and BILCC are similar to each other but their computational complexities may be different from the viewpoint of limited use of workspace. BILCC seems to be harder than LCCR because of its additional constraint on the output order (as noted in Table 1). But it is not known whether it is true.

Among the five basic tasks or problems, Connected Components Labeling has been extensively studied and a number of algorithms have been proposed so far under several different computational models including pipelining, parallel computation [1,4,7,8,10–12,16–20]. A nice survey [13] on this topics is also available. Since we may have $O(n)$ connected components for a binary image of n pixels, the whole label matrix takes $O(n \log n)$ bits in total. If $O(n)$ work space is available, it is not so hard to design a linear-time algorithm for connected components labeling.

Please cite this article in press as: T. Asano, et al., A new algorithmic framework for basic problems on binary images, Discrete Applied Mathematics (2016), http://dx.doi.org/10.1016/j.dam.2016.02.025