알고리즘 요해: 사례를 중심으로

디지털 시대가 요구하는 데이터와 인공지능 거버넌스 체재를 모색하기 위해서는 알고리즘과 데이터의 기술적, 사업적 현황에 관한 정확한 이해가 먼저 필요하다. DAIG의 프라이머(Primer) 시리즈는 올바른 거버넌스 논의에 필요한 기초를 제공할 목적으로 기획 연재된다. 첫 프라이머로 온라인 소비자들이 시장에서 흔히 접하는 알고리즘의 유형과 활용에 관하여 살펴본다.

I. '알고리즘' 사회

II. 알고리즘의 의미

1. 정의
2. 알고리즘 적용의 전제 조건
3. 예시: 정렬 알고리즘

III. 알고리즘의 효율성과 정확성

1. 알고리즘의 효율성
2. 알고리즘 정확성

IV. 검색 알고리즘의 사례

1. 검색 엔진의 일반적 작동 원리
2. 페이지랭크 알고리즘
3. 다양한 순위 결정 알고리즘의 혼합 적용

V. 결론

알고리즘의 의미와 사례

구본효
서울대학교 법학대학원
박사과정

김병필
KAIST 기술경영학부
초빙교수·변호사

I. '알고리즘' 사회

'알고리즘'[1]이라는 표현은 이제 일상에서도 자주 접하게 되었다. 검색 엔진 사이트가 검색 알고리즘을 '조작'하여 자사 서비스를 우대하는 행위가 국내외적으로 비판받고 있다. 포탈 사이트는 기사의 순위나 배치가 문제시되는 경우 알고리즘에 의해 자동적으로 결정되므로 뉴스 서비스에 자의성이 개입될 여지가 없다고 항변한다. 동영상 제공 서비스는 이용자들이 사이트에 최대한 오래 머무르도록 추천 알고리즘을 이용하여 개인 맞춤형 영상을 제시한다. 콘텐츠 제공 플랫폼은 저작권 침해 여부 등 위법 여부를 자동적으로 탐지하는 콘텐츠 모더레이션(contents moderation) 알고리즘을 사용하기도 한다. 알고리즘 트레이딩은 이미 매우 널리 활용되고 있는 투자 기법이며, 그 외에도 보안 또는 암호화 알고리즘, 채용이나 직원 평가 알고리즘, 신용 평가나 대출 심사 알고리즘 등 알고리즘이 활용되는 예시는 매우 다양하다.

최근 들어 알고리즘은 인공지능과 유의어처럼 사용되기도 한다. 앞서 든 예시들에서 '알고리즘' 대신 '인공지능'으로 표현을 바꾸더라도 어색하지 않다. 그렇다면 인공지능과 알고리즘의 관계는 어떻게 되는 것일까? 우선 '알고리즘'이 '인공지능'을 포함하는 상위 개념으로 사용되는 경우가 많다. 정보처리 과정을 자동화하는 방식을 뭉뚱그려 알고리즘이라 표현하기도 한다. 알고리즘은 딥 러닝(deep learning) 등 최신 인공지능 기술을 활용하는 경우만을 지칭하는 것이 아니다. 가령 대출 심사 알고리즘은, ① 인공지능 기법(특히 머신러닝)을 활용하여 채무불이행 위험성을 예측하는 경우뿐만 아니라, ② 금융기관이 일정한 규칙을 미리 정하여 대출 신청자를 스크리닝하는 경우를 포함한다.

이처럼 알고리즘은 흔히 인공지능의 상위 개념으로 이해되지만, 관점에 따라서는 알고리즘이 인공지능의 하위 구성 요소에 해당하기도 한다. 예컨대 머신러닝 방식의 인공지능의 구성 요소는 ① 데이터를 학습하는 알고리즘과 ② 학습 대상인 데이터로 구분된다. 만약 인공지능이 오동작한다면, 이는 학습 알고리즘에 문제가 있을 수도 있고, 학습 데이터에 문제가 있을 수도 있다.

이처럼 알고리즘이라는 용어는 폭넓게 활용되고 있지만, 그 단어가 지칭하는 의미의 범위는 명확하지 않고, 이를 확실하게 정의하기도 어렵다. 그러나 '알고리즘'의 의미를 정확하게 이해하는 것이 더욱 중요해지고 있다. 이는 무엇보다도 '알고리즘'이 점차 법적 규율의 대상으로 인식되고 있기 때문이다. 예컨대 기사 배열 알고리즘이나 추천 알고리즘에 있어 자의성이나 편향성을 막는 것은 민주적 여론 형성을 위해 긴요하다. 평등권 실현을 위해서는 채용 알고리즘의 공정성을 보장할 필요도 있다. 이러한 맥락에서 '알고리즘'은 흔히 규제의 대상으로 이해된다. 이와 같은 알고리즘에 대한 규율 방안이 생산적으로 논의되기 위하여 알고리즘에 관한 보다 정확한 이해가 선행될 필요가 있다.

---

1)외래어 표기법을 엄격히 따르자면 'algorithm'의 올바른 한글 표기법은 '알고리듬'이 되지만, '알고리즘'이 더 널리 사용되고 있다. 한편 '알고리즘'은 아라비아 숫자를 사용하는 연산을 의미하는 'algorism'의 한글 표기법이기도 하다(외래어 표기법 제3장 제1절 제3항). 표준국어대사전은 알고리듬(algorithm)의 의미와 알고리즘(algorism)의 의미가 동일하다고 한다. 알고리즘은 이미 굳어진 외래어라고 볼 수도 있겠다. 참고로 지능정보화 기본법 제60조 제1항 제4호는 "알고리즘의 제공에 관한 사항"이라 규정하여, 이것을 뒷받침하고 있다. 그러므로 이 글에서는 알고리즘이라고 적는다. Merriam-Webster, "algorism", https://www.merriam-webster.com/dictionary/algorism (2020. 9. 6. 확인).

II. 알고리즘의 의미

1. 정의

컴퓨터 과학 분야에서 널리 사용되고 있는 교과서에서는 알고리즘을 다음과 같이 설명하고 있다. "알고리즘은 어떤 값(들)을 입력받아 어떤 값(들)을 출력으로 생성하는 잘 정의된 계산 절차이다. [...] 알고리즘은 잘 명시된 계산 문제 해결을 위한 도구라고도 볼 수 있다."[2] 이러한 개념 설명에서 핵심적 요소는 입력, 출력, 계산 절차이다. 컴퓨터가 무엇인가 처리를 하기 위해서는 어떠한 값을 입력(input)받아야 한다. 그 입력으로부터 처리되어 나오는 것이 출력(output)이다. 이러한 관계는 계산 절차(computational procedure)를 통해 정해진다.[3]

따라서 알고리즘을 이해하기 위한 첫 번째 작업은 입력과 출력이 무엇인지 이해하는 것이다. 아래에서 소개하는 숫자 정렬 알고리즘과 같은 단순한 사례에서는 입력과 출력이 명확하다. 뒤섞인 숫자의 리스트가 입력이고, 순서대로 정렬된 숫자의 리스트가 출력이다. 하지만 현실적으로 활용되는 많은 알고리즘에 있어서는 입력과 출력을 엄밀히 정의하는 작업이 간단하지 않다.

가령 검색 알고리즘을 생각해보자. 여기서 입력은 이용자가 입력한 검색어(query)이고, 출력은 해당 검색어에 비추어 관련된 중요성이 높은 웹 문서들의 목록이다. 여기서 출력 결과인 '관련된 중요성이 높은 웹 문서들'은 무슨 의미인가? 이용자가 입력한 단어가 많이 포함되어 있을수록 중요성이 높다고 볼 수도 있고, 다른 이용자들이 더 자주 클릭하는 문서가 중요성이 더 높다고 생각할 수도 있다. 제4장에서 설명할 페이지랭크(PageRank) 알고리즘과 같이 다른 문서들에 의해 더 많이 인용(링크)되는 문서가 중요도가 높다고 생각할 수도 있다. 따라서 검색 알고리즘에서는 '중요도'를 어떻게 정의하는지에 따라 출력 및 계산 절차가 크게 달라진다.

한편, 넓은 의미에서 보면 반드시 컴퓨터만 알고리즘을 활용하는 것은 아니다. 컴퓨터가 존재하기도 전부터 사람들은 알고리즘을 활용하고 있었다. 예컨대 고대 그리스의 수학 교본으로 알려진 유클리드의 '원론'에는 최대공약수를 구하는 알고리즘이 소개되어 있다. 이는 유클리드 호제법이라고 하며, 오늘날까지 내려오는 가장 오래된 알고리즘으로 알려져 있다.[4] 이 방법은 자연수를 다른 자연수로 나누어 그 나머지를 구하는 과정을 반복함으로써 최대공약수를 구하는 것이다. 이러한 방법도 입력(두 개의 수)과 출력(최대공약수) 간의 관계를 정해 놓은 계산 방식(나눗셈의 반복)이므로

---

2) Thomas H. Cormen et al., Introduction to algorithms, MIT Press, 3rd ed. 2009. "Informally, an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. [...] We can also view an algorithm a tool for solving a well-specified computational problem."

3) 알고리즘에 대해 탐구하고자 하는 독자에게 다음의 도서와 논문을 추천한다. Thomas H. Cormen et al., supra note 2. Introduction to Algorithms는 가장 일반적으로 사용되는 교과서이며, 저자들의 성을 축약하여 CLRS(Cormen, Leiserson, Rivest, Stein)라고도 말한다. 그보다는 간단한 내용으로 다음의 책이 유용하다. Thomas H. Cormen, Algorithms Unlocked (2013). Panos Louridas, Algorithms (2020). Algorithms는 쉽고 간단하면서도, 최근에 연구되고 있는 인공지능의 방법론(deep learning)까지도 소개하고 있다. John MacCormick, Nine algorithms that changed the future: the ingenious ideas that drive today's computers (2012). 우리가 일상적으로 사용하고 있는 알고리즘에 대한 설명으로는 위의 미래를 바꾼 아홉 가지 알고리즘을 참조할 수 있다. 암호, 인공지능, 데이터베이스, 디지털 서명 등 '위대한' 알고리즘이 소개된다.

4) Donald Knuth, The Art of Computer Programming, Addison-Wesley, 3rd ed. 1997.

알고리즘이라고 볼 수 있다.

요컨대, 알고리즘은 문제를 해결하기 위해 입력과 출력과의 관계를 정해 놓은 계산 절차로 이해할 수 있다.

2. 알고리즘 적용의 전제 조건

어떤 문제를 해결하기 위한 도구로서 알고리즘을 적용하려면, 그 문제가 잘 명시되어(well-specified) 있어야 한다. 즉, 문제를 명시하는 것은 알고리즘의 전제 조건에 해당한다. 따라서 어떤 문제를 알고리즘 방식으로 해결하기 위해서는 우선 주어진 문제를 명시적으로 기술된 계산 문제로 환원해야 한다.[5)]

하지만, 최근에는 정확하게 정의하기 어려운 문제에 대해서도 알고리즘을 활용하고자 하는 시도가 늘어나고 있다. 예컨대 2018년 정부는 '지능정보형 인사정책지원' 플랫폼을 도입한다고 발표했다.[6)] 이러한 플랫폼은 특정 직위에 필요한 직무요건을 생성하고 그 직무요건에 '적합한 인물'을 추천한다고 한다. 보도자료만 보아서는 해당 알고리즘이 '적합한 인물'을 어떻게 정의하고 있는지 알기 어렵다. 하지만 알고리즘을 적용했다는 것은 여하한 형태로든 적합성의 의미가 무엇인지 정의되어 있다는 것을 뜻한다. 따라서 이러한 인사 추천 알고리즘을 이해하고 평가하기 위해서는 '적합한 인물'이 어떻게 정의되어 있는지를 규명하는 것이 중요해진다. 이러한 문제는 아래 제3장(알고리즘의 효율성과 정확성)에서 더욱 상세하게 다룬다.

3. 예시: 정렬 알고리즘

대학교 알고리즘 과목에서 흔히 가장 먼저 접하는 알고리즘은 정렬 알고리즘(sorting algorithm)이다. 이는 임의의 순서로 나열된 숫자를 미리 정해진 순서대로 늘어서도록 하는 것이다. 가령 엑셀 프로그램의 정렬 기능을 이용하면 수 만 개의 데이터도 손쉽게 정렬되는 것을 경험할 수 있다. 이 경우 알고리즘의 입력은 임의로 뒤섞인 숫자들의 목록이고, 그 출력은 정해진 순서대로 정렬된 목록이다.[7)] 문제는 정렬하는 방법이다.

가장 단순한 방법은 이러하다. 가령 (5, 2, 4, 1, 3)이라는 숫자의 목록을 작은 숫자가 앞에 오도록 정렬하는 경우를 생각해보자. 우선 첫 숫자 5와 둘째 숫자 2를 비교한

---

5)반대로 알고리즘이 휴리스틱(heuristic)으로 해법을 발견하는 것도 가능하다. 알고리즘이 해결하는 문제는 대표적으로 최선의 답을 발견하는 최적화(optimization problem), 예 또는 아니오로 답하는 결정(decision problem) 문제가 있다. 최적화는 머신러닝(machine learning)을 구성하는 주요한 알고리즘이다. 기계학습은 인간이 경험으로부터 학습하는 능력을 부분적으로 구현하는 인공지능의 방법론이라고 한다. 그러므로 기계학습의 특성은 데이터로부터 지식을 습득한다는 것이다. 이는 사전적으로 입력된 지식(hard-coded knowledge) 또는 논리적으로 추론하는 규칙(logical inference rule)으로부터 자동으로 판단하는 시스템(knowledge based system), 구체적으로, 전문가 시스템(expert system)과는 차이가 있다. 일반적으로 머신러닝 알고리즘은 지식을 수학적으로 표상한다(mathematical model). 다음으로 학습이 이루어지는 정도를 나타내는 함수를 정의한다(objective function). 최적화 알고리즘은 그것을 최대화하는 변수를 발견하기 위하여 사용되는데, 확률적 경사 하강법(stochastic gradient descent) 등이 대표적이다.

6)인사혁신처, AI가 자리에 맞는 적합한 인재 찾아낸다, 2018. 12. 24.

7)순서를 정의하는 방법도 다양할 수 있다. 일반적으로 숫자의 경우는 그 크기에 따라 오름차순과 내림차순으로 정렬할 수 있다. 하지만 문자의 경우에는 다양한 방식의 순서를 정의할 수 있다. 가령 옥편에서 한자를 순서대로 배열하는 방법은 매우 다양하다.

다. 2가 더 작으므로 둘의 순서를 바꾼다. 그러면 (2, 5)가 된다. 앞 두 숫자의 비교가 끝났으니 이제 셋째 숫자인 4를 본다. 이미 앞선 목록과 비교해서 순서상 들어맞는 위치를 확인한다. 그러면 2와 5의 사이에 4를 끼워 넣으면 된다. 이제 (2, 4, 5)가 되었다. 다음으로 넷째 숫자인 1을 가장 첫 자리에 삽입하면 (1, 2, 4, 5)가 된다. 마지막 숫자인 3을 알맞은 위치에 삽입하면 (1, 2, 3, 4, 5)가 된다(그림 1).[8] 이러한 방법은 숫자를 차례대로 알맞은 위치에 삽입하는 것이므로, '삽입 정렬(insertion sort)'이라 한다. 삽입 정렬 알고리즘은 직관적으로 이해하기 쉽고 구현하기도 쉽다. 그러나 정렬하고자 하는 숫자가 많아질수록 실행 속도가 크게 느려진다는 단점이 있다.

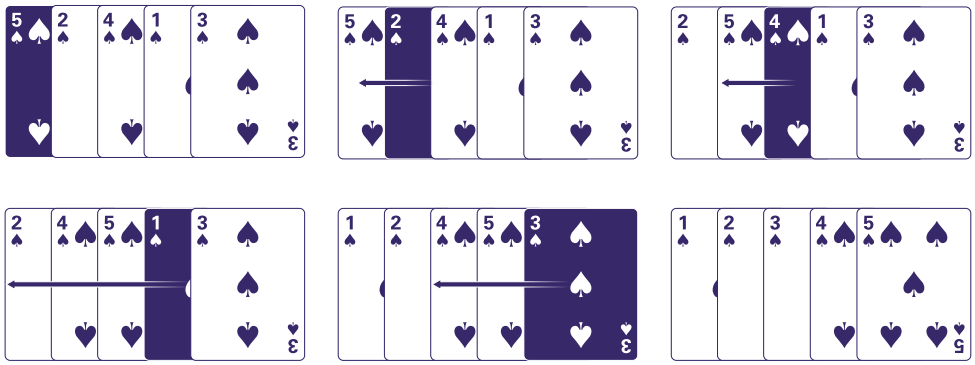

그림 1. 카드 정렬하기(삽입 정렬)

그렇다면 더 효율적으로 실행되는 알고리즘은 없을까? 지금까지 발견된 정렬 알고리즘은 수십 개 이상이다. 예컨대 현대 컴퓨터의 아버지이자 20세기 최고의 천재 중 한 명으로 손꼽히는 요한 폰 노이만(Johann von Neumann)은 1945년 더 효율적인 정렬 방법을 제시했다. 그가 제시한 방법은 이러하다. 정렬 대상 숫자들을 계속 두 묶음(부분열)으로 나눈다. 다음으로 각각의 부분열을 미리 정해진 순서대로 늘어서도록 처리한다. 마지막으로 부분열을 다시 병합한다(그림 2).[9] 그런데 각각의 부분열을 다시 병합하는 경우에 최솟값만을 비교하여 작은 숫자부터 나열하여도 된다. 왜냐하면 각각의 부분열이 이미 순서대로 정렬되었기 때문이다.[10] 이것을 '병합 정렬(merge sort)'이라 한다.[11] 직관적으로 이해하기는 어렵지만, 병합 정렬은 삽입 정렬에 비해 훨씬 효율적이다.

이처럼 비교적 간단한 정렬 알고리즘을 통해서 우리는 다음과 같은 점을 확인할 수 있다. ① 알고리즘을 통해서 해결하고자 하는 문제가 잘 명시되어 있다. 정렬 알고리즘이 해결하는 문제는 뒤섞인 숫자 목록의 순서를 정렬하는 것이다. ② 알고리즘의

8) Thomas H. Cormen et al., 앞의 책(주 2).

9) Thomas H. Cormen et al., 앞의 책(주 2).

10) Thomas H. Cormen et al., 앞의 책(주 2).

11) 이것은 소위 분할정복법(divide-and-conquer)이라고 하는 사고방식이 적용된 알고리즘이다. 분할정복법은 문제를 비슷하면서도 작은 문제들로 분할(divide)한 다음 그것을 재귀적으로 풀어(conquer) 원래대로 합하는(combine) 사고방식이다. Thomas H. Cormen et al., 앞의 책(주 2).

입력 값과 출력 값이 명확하게 주어져 있다. 입력 값은 뒤섞인 숫자 목록이고, 출력 값은 정렬된 숫자 목록이다. ③ 주어진 입력 값으로부터 출력 값을 얻기 위한 계산 절차에는 여러 방법이 있을 수 있다. 삽입 정렬과 같이 직관적이고 이해하기 쉬운 방법도 있지만, 병합 정렬과 같이 곧바로 이해하기 어려운 방법도 있다.

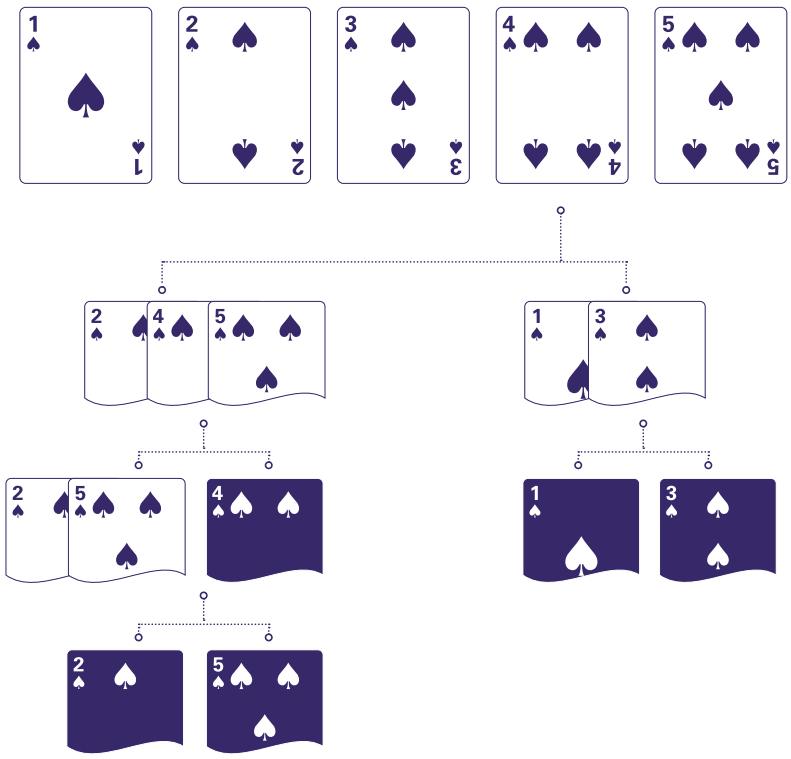

그림 2. 카드 정렬하기(병합 정렬)

III. 알고리즘의 효율성과 정확성

이처럼 같은 문제를 해결하는 알고리즘이 여러 개가 존재한다면, 어떤 알고리즘이 더 우수하다고 평가할 수 있을까? 일차적으로는 효율성이 중시된다. 같은 문제를 최대한 신속하게 적은 계산 자원을 이용하여 해결하는 알고리즘이 선호되는 것은 당연하다. 컴퓨터 과학자들이 여러 알고리즘 중 무엇이 가장 효율적인지 비교하는 작업을 알고리즘 분석(analysis of algorithm)이라 한다. 일반적인 기준은 알고리즘이 종료되는 데 걸리는 시간과 알고리즘이 사용하는 메모리 공간의 크기이다.

물론 알고리즘이 해결하고자 하는 과제를 정확하게 해결해 내는지 검증하는 것도 중요하다. 예컨대 내비게이션 소프트웨어의 경로 탐색 알고리즘을 생각해보자. 경로 탐색 알고리즘은 '최소 시간', '최단 거리', '최소 비용' 등의 조건을 충족시키는 경로를

도출한다. 소프트웨어 개발자는 알고리즘이 이러한 조건을 정확하게 만족시키는지 테스트해야 한다. 하지만 알고리즘 자체는 정확하더라도 이를 구현하는 단계에서 의도하지 않은 오류가 존재하게 되는 경우도 발생할 수 있다. 프로그래머의 부주의로 오류가 개입되는 때도 있을 수 있고(소프트웨어 '버그'가 있는 경우), 입력된 데이터 자체에 오류가 있는 때(가령 내비게이션 소프트웨어의 지도 자체나 시간별 교통량 정보가 잘못된 경우)도 있을 것이다. 따라서 이러한 다양한 경우를 고려하여 알고리즘을 테스트하는 방법이 필요하게 된다, 이하에서는 알고리즘에 대한 효율성과 정확성 평가에 관해 차례로 살펴본다.

1. 알고리즘의 효율성

알고리즘이 실행되는 동안 소요되는 연산 자원(computational resources)이 적을수록 알고리즘이 효율적이라고 한다. 그 기준은 알고리즘이 종료될 때까지 걸리는 시간과 알고리즘이 사용하는 메모리 용량이다. 알고리즘이 더 자주 사용될수록, 알고리즘이 실행되는 컴퓨터 하드웨어의 성능이 낮을수록(가령 스마트폰 단말기) 효율성은 더욱 중요한 요소가 된다.

알고리즘의 수행 시간은 '시간 복잡도(time complexity)'라는 개념을 통해 평가한다. 알고리즘의 수행 시간은 입력되는 자료의 크기에 따라 달라진다. 입력되는 자료의 양이 늘어나면 처리 시간도 더 오래 걸리는 것은 당연하다. 그런데 처리 데이터가 증가하면 수행 시간은 그보다 훨씬 더 많이 늘어나는 경우가 많다. 예컨대 앞서 설명한 삽입 정렬 알고리즘의 소요 시간은 입력된 숫자 개수의 제곱에 비례하여 늘어난다. 즉, 숫자 1,000개를 정렬하는 데 0.01초가 걸렸다고 하면, 숫자 2,000개를 정렬하기 위해서는 그 2배가 아니라 4배인 0.04초가 걸리게 된다. 숫자를 서로 비교해야 하는 횟수가 이에 비례해서 증가하기 때문이다. 만약 정렬해야 할 데이터가 10배로 증가하면 수행 시간은 그 제곱인 100배로, 데이터가 100배 증가하면 수행 시간은 그 제곱인 10,000배로 증가한다.

이를 두고 삽입 정렬 알고리즘의 시간 복잡도는 $O(n^2)$이라고 표현한다. 알파벳 대문자 O를 사용하므로 빅오 표기법(Big-O notation)이라도 부른다. 알고리즘의 시간 복잡도가 $O(n^2)$이라면 처리 데이터가 많은 경우에는 활용이 어려울 수도 있다.

이에 비해 병합 정렬의 경우 실행되는 데 소요되는 시간은 에 비례하여 늘어난다.[12] 즉, 병합 정렬 알고리즘의 시간 복잡도는 O(nlogn)이다. 정렬 데이터가 100배 늘어나면 정렬 시간은 인 약 664배가 증가한다. 따라서 병합 정렬이 삽입 정렬에 비해 훨씬 효율적이라고 한다.[13]

시간 복잡도가 $O(n^2)$인 알고리즘과 O(nlogn)인 알고리즘을 비교하면 <그림 3>과 같다. 이 그래프를 보면 처리해야 하는 자료의 양이 증가할수록 두 알고리즘 간의 성능 차이가 급격하게 벌어진다는 점을 알 수 있다. 이처럼 같은 입력을 받아 같은 출력을

12)여기서 가 나오는 이유는 병합 정렬의 경우 정렬 대상 숫자열을 재귀적으로 절반으로 나누어 처리하기 때문이다.
13)Thomas H. Cormen et al., 앞의 책(주 2).

생성해 내는 알고리즘이라도 하더라도 처리 방법에 따라 소요 시간이 크게 달라질 수 있다.

알고리즘의 효율성은 제도적 규율로부터 비교적 자유롭다. 일반적으로 효율성을 증가시키는 것은 컴퓨터 과학이나 공학의 문제로 취급된다. 컴퓨터 과학의 역사에는 종래 수 주일이나 수개월이 걸렸던 문제를 알고리즘의 효율성을 개선하여 몇 시간, 몇 분 이내에 처리할 수 있게 된 사례가 많다. 최근의 인공지능 기술의 성장에 있어서도 알고리즘의 효율성 개선이 기여한 바가 크다. 매우 복잡한 인공신경망(artificial neural network)을 학습시키는 핵심 알고리즘의 성능이 최근 10-100배가량 개선되면서 인공지능의 새로운 가능성이 열린 것으로 평가된다.[14)]

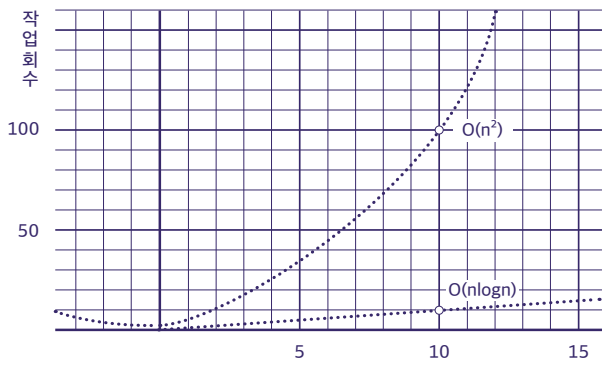

그림 3. 알고리즘의 시간 복잡도에 따른 작업 횟수

2. 알고리즘 정확성

(1) 알고리즘의 사양(specification)과 정확성

알고리즘이 주어진 문제를 해결한다면 그 알고리즘은 정확하다(correct)고 한다. 알고리즘이 정확하면 언제나 올바른 값이 출력된다. 앞서 소개한 정렬 알고리즘은 모두 정확한데, 이는 항상 순서대로 정렬된 숫자 목록이 출력된다는 뜻이다. 알고리즘의 정확성을 평가하려면 알고리즘에서 요구되는 기능이 미리 정의되어야 한다. 그것을 사양(specification)이라 한다. 알고리즘의 사양은 분명하고도 완전한 내용으로 기술되어야 하고, 중의적으로 해석될 가능성이 없어야 한다. 그리고 알고리즘에 요구되는 조건이 전부 포함되어야 한다.

따라서 알고리즘의 정확성 평가는 두 가지 차원으로 구분된다. 만약 소프트웨어가 정확하지 못하다면 크게 보아 두 가지 가능성이 있을 수 있기 때문이다. ① 우선, 알고리즘이 사양을 충족시키지 못하는 경우("software failing to meet its specification")가 있을 수 있다. ② 두 번째로는 사양 자체가 잘못 설계된 경우("software faithfully implementing an incorrect specification")가 있을 수 있다. 이하에서는 이를 차례대로 살펴본다.

14)Eric Brynjolfsson and Andrew McAfee, "What's driving the machine learning explosion?", Harvard Business Review 2017. 7. 18.

(2) 알고리즘의 사양 충족 여부 테스트

소프트웨어에 대한 테스트는 프로그래머가 알고리즘의 사양을 제대로 구현했는지를 시험하는 경우가 대부분이다. 이러한 시험은 알고리즘이 실행되기 이전에 코드만으로 평가하는 시험(static testing)과, 알고리즘이 실행된 후 결과를 평가하는 시험(dynamic testing)으로 구분된다. 그리고 소프트웨어의 구조가 공개되는 시험(white-box testing)과 비공개되는 시험(black-box testing)으로도 구분된다.[15)]

알고리즘이 정확하게 구현되기 위해서는 프로그래머가 오작동의 원인을 부주의하게 삽입하지 아니하여야 한다. 알고리즘 구현 과정에서 야기된 오류를 흔히 소프트웨어 '버그(bug)'라고 부른다. 이러한 용어는 예전 진공관으로 된 거대한 컴퓨터에 벌레가 들어가서 프로그램이 오동작한 사례에서 유래한 것으로 알려져 있다. 특히 항공우주나 의료 분야에 적용되는 소프트웨어에 있어서는 '버그'가 없도록 보장하는 것이 무엇보다 중요하다. 이처럼 알고리즘 구현 단계에서의 정확도를 보장하는 것은 '소프트웨어 공학'이라는 독립된 학문 분야가 있을 정도로 방대하고 중요한 분야다.

하지만 알고리즘의 구현 단계에서 오류가 없다고 하더라도 알고리즘의 출력이 정확하리라고 보장할 수 없다. 가령 내비게이션 알고리즘이 주어진 조건에 맞추어 경로를 도출해 낸다고 하더라도, 만약 부정확한 지도나 교통 정보가 주어졌다면 그로부터 출력된 경로는 정확하지 않다. 즉, 알고리즘이 사양대로 작동하도록 구현된 경우에도 올바른 값이 출력되지는 않을 수 있다.[16)] 따라서 알고리즘의 정확도에 대한 평가는 구현 상의 오류가 없는지 뿐만 아니라 입력 값의 정확성에 대한 평가를 포함하는 작업이 된다.

(3) 알고리즘의 사양 자체에 대한 테스트

다음으로 애당초 사양이 잘못된 설계된 것이 아닌지를 시험할 필요가 있다. 이는 쉽지 않은 문제이다. 내비게이션 소프트웨어의 경로 탐색 알고리즘의 예를 다시 생각해보자. '최소 시간', '최단 거리', '최소 비용' 등의 조건이 사양으로 주어진 경우, 이러한 조건은 비교적 이해하기 간명하고, 중의적으로 해석될 가능성이 작다. 하지만 이용자가 선택할 수 있는 옵션에는 "최적 경로"라는 조건도 있다. 무엇이 '최적'인가? 넓고 운전하기 쉬운 길을 가는 경로와 좁은 골목길을 거쳐 가는 경로 중 어느 쪽이 최적인가? 이면도로와 같이 사고 위험이 큰 길은 가급적 회피하는 것이 최적일 수도 있고, 출퇴근 시간에는 이러한 길을 통해서라도 시간을 줄이는 것이 최적일 수 있다. 다른 한편 모든 내비게이션 이용자들이 같은 소프트웨어를 이용하여 같은 추천 경로를 이용한다면 오히려 그로 인해 예상 시간보다 지연되는 부정적 피드백 효과가 발생할 수도 있다. 이러한 상황도 고려해야 최적이라고 할 수 있다. 이처럼 '최적' 조건을 설정하기 위해서 결정해야 할 요소는 매우 많다.

---

15) Joshua A. Kroll et al., "Accountable Algorithms," 165 University of Pennsylvania Law Review 633-705 (2017).

16) Deven R Desai & Joshua A Kroll, "Trust But Verify: A Guide to Algorithms and the Law," 31 Harvard Journal of Law & Technology 1-64 (2018).

비록 알고리즘이 종국적으로 달성하고자 하는 목적이 불명확한 경우라 하더라도 알고리즘에 주어지는 사양에는 알고리즘으로 구현할 조건이 무엇인지 명시되어 있어야 한다. 가령 내비게이션 알고리즘은 아마도 이동 시간, 운전의 편의성, 안전성 등의 여러 요소마다 서로 다른 가중치를 두고 이를 종합하여 '최적' 경로를 도출할 것이다.

이처럼 소프트웨어의 사양을 결정하는 과정을 흔히 '설계(design)'라 부른다. 설계 과정에는 의식적으로든 아니든 사회적 가치에 관한 판단이 반영된다. 콘텐츠 추천 알고리즘을 예로 들어 보자. 만약 설계 단계에서 이용자가 클릭할 가능성이 가장 큰 콘텐츠를 추천하는 것으로 알고리즘의 사양이 정해졌다고 하자. 그러면 이용자가 콘텐츠 플랫폼에 체류하는 시간은 길어질 수 있고, 그 결과 플랫폼의 광고 수입은 더 증가할 수 있다. 하지만, 그만큼 이용자가 다양한 콘텐츠를 접할 가능성이 작아지는 반대 효과가 발생할 수도 있다. 그 결과 필터 버블(filter bubble)이 형성되고, 집단 극화(group polarization) 현상이 나타날 위험이 생겨날 수 있다.

그래서 알고리즘이 사회적으로 미치는 영향을 평가해야 한다는 주장도 제기된다. 이는 특히 알고리즘에 관한 법·제도적 규율의 맥락에서 중요하다. 아마존(Amazon)은 2014년부터 고용 인공지능을 개발하였는데, 남성 지원자가 더 유리하다는 사실을 실험으로부터 확인하자 곧바로 프로젝트를 중단하였다.[17] 또한 미국 형사 사법절차에서 활용되고 있는 피고인의 재범 가능성 예측 소프트웨어에 대해 유색인종을 차별한다는 비판이 제기되기도 하였다.[18]

결국 알고리즘에 관한 법적 규율은 알고리즘의 사양을 정해주는 작업과 유사하다. 예를 들어 개인정보의 안전성 확보조치 기준은 일정한 개인정보를 '안전한 암호 알고리즘'으로 암호화하여 저장할 것을 요구한다(동 기준 제7조 제5항). 여기서 '안전한 암호 알고리즘'은 국내외의 연구 기관이 권고하는 것을 의미하는데, 개인정보의 암호화 조치 안내서는 그 예시로 AES, SHA 등의 방식을 제시한다. 이 경우 알고리즘의 사양은 국내외의 연구 기관이 권고하는 암호 알고리즘이 될 것이다.

이렇게 놓고 보면 앞서 예로 든 인공지능에 의한 차별 문제는 법이 분명하고 완전한 사양을 제공하지 못하고 있어 발생하는 문제라고 생각할 수도 있다. 그러나 법을 통해 알고리즘 개발자에게 공정성에 대한 분명하고도 완전한 사양서(requirements specification)를 제공하는 것이 완전한 해결책이 되기 어렵다. 반대로 법 제도가 이처럼 분명하고도 완전한 내용으로 사양을 제공할 수 있는 경우는 오히려 드물다. 법 규범의 문언에는 가치 판단을 포함한 일반적, 규범적 개념이 사용된다. 특히 알고리즘과 같이 규율 대상이 지극히 다양하거나 수시로 변화하는 성질의 것이어서 일의적으로 규정할 수 없는 경우에는 명확성 요건을 엄격하게 요구하기도 어렵다.[19] 그러므로 법 규범이 알고리즘에 대한 명확한 사양으로 기능하기는 쉽지 않은 경우가 많을 것이다.[20]

---

17) Reuters, 2018. 10. 11., "Amazon scraps secret AI recruiting tool that showed bias against women."

18) ProPublica, 2016. 5. 23., "Machine Bias".

19)헌법재판소 1999. 9. 16. 선고 97헌바73 결정.

20) Danielle Keats Citron, "Technological Due Process," 85 WASH. U. L. REV. 1249 (2008)

IV. 검색 알고리즘의 사례

1. 검색 엔진의 일반적 작동 원리

본 장에서는 알고리즘의 중요한 적용 사례로 구글 검색 서비스의 기초가 된 페이지랭크(PageRank) 알고리즘을 살펴본다. 일반적으로 인터넷 검색 엔진이 작동하는 원리는 대략 다음과 같다. 우선 웹상으로 업로드된 웹 페이지의 내용을 수집한다. 이를 크롤링(crawling)이라고 한다. 크롤러는 수천 억 개에 달하는 웹 페이지로부터 정보를 주기적으로 가져온다.[21] 다음으로 그 내용을 확인하며 데이터베이스에 저장한다. 이를 색인(indexing)이라고 한다. 마지막으로 이용자가 검색어를 입력하면, 색인으로부터 검색어와의 관련성이 있고 중요하다고 생각되는 것을 분석하여 이용자에게 표시한다. 이를 두고 검색(searching)이라고 한다.

흔히 인터넷 검색 알고리즘은 그저 관련성이 있는 웹페이지를 찾는 과정으로 생각하기 쉽다. 하지만 검색 알고리즘에서 훨씬 더 중요한 것은 이용자 화면에 표시되는 순서를 결정하는 것이다. 관련성이 있는 웹 페이지를 찾는 과정을 매칭(matching)이라고 하고, 그 순위를 결정하는 과정을 랭킹(ranking)이라고 한다. 매칭 과정에서 수만, 수십만 개, 혹은 그 이상의 웹 페이지가 찾아질 수 있으므로, 이를 잘 선별하여 보여주는 것이 검색 서비스의 핵심적 기능이 된다.

페이지랭크 알고리즘은 그 이름에서 파악할 수 있듯이 주어진 검색어와 매칭된 웹 페이지의 관련된 중요도를 정하여 그 순위를 결정하는 알고리즘이다. 이는 흔히 현재의 구글이 있게 한 중요한 알고리즘으로 평가된다.

2. 페이지랭크 알고리즘

(1) 배경

페이지랭크 알고리즘은 이용자에게 유의미하고 중요한 정보가 무엇인지 객관적으로 평가하는 측도(measure)를 제시한다. 검색 서비스는 어마어마한 양의 웹 페이지 중에서 이용자가 관심이 있는 웹 페이지가 무엇인지 옥석을 가려내어야 한다. 페이지랭크 알고리즘은 이를 성공적으로 수행한다. 구글의 창업자 세르게이 브린(Sergei Brin) 그리고 래리 페이지(Larry Page)의 저명한 논문은 그 알고리즘을 처음으로 소개하였다.[22]

논문이 작성되었을 당시의 주된 검색 엔진 서비스였던 야후(Yahoo), 알타비스타(Altavista) 등이 사람이 관리하는 색인 형식으로 검색을 운영하였다. 이는 주관적이기도 하였고 만만찮은 비용이 수반되기도 하였다고 한다. 대다수의 관심을 받는 주제는 효과적으로 포함하는 반면에, 소수의 몇 사람만이 관심을 기울이는 주제는 포함하지 아니하였다. 그러므로 웹상 정보가 기하급수적으로 성장하고 있는 상황을 고려하

---

21) Google 검색, "검색에서 정보를 구성하는 방법", https://www.google.com/intl/ko/search/howsearchworks/crawling-indexing/ (2020. 9. 6. 확인).

22) Sergey Brin & Lawrence Page, "The anatomy of a large-scale hypertextual Web search engine," 30 Computer Networks and ISDN Systems 107-117 (1998).

면, 기존 시스템으로는 이를 제대로 처리하는 것이 불가능하였다.

한편, 하나의 검색어에 대응되는 색인에도 방대한 양의 문서가 포함되어 있고, 계속적으로 증가하고 있었다. 반면, 사용자가 관심이 있는 문서는 그 중 고작 수십 개 정도에 불과하다. 그래서 기존의 검색 서비스가 제공하고 있던 '키워드 매칭(keyword matching)' 방식만을 적용하면 이용자에게 유의미하고 중요한 정보가 제공되기 어려웠다. 또한 일부 광고주는 이것을 남용하여 이용자에게 중요한 정보가 아니라 광고주에게 유리한 결과를 표시하는 사례도 있었다. 이러한 배경에서 브린과 페이지는 가장 중요하고도 유의미한 정보가 무엇인지 객관적으로 평가하는 측도가 요구된다고 생각하였다.

(2)페이지랭크 알고리즘의 개요

페이지랭크 알고리즘은 인터넷 상의 웹 페이지가 하이퍼링크(hyperlink)를 사용하여 다른 웹 페이지와 연결되어 있다는 점에서 착안한 것이다. 웹 페이지 상에서 보통 밑줄 친 파란색 글씨로 표시되는 하이퍼링크는 다른 웹 페이지로의 이동을 가능하게 한다. 이렇게 보면 하이퍼링크는 과학 논문에서 다른 논문을 인용하는 것과 유사하다고 생각할 수 있다. 일반적으로 과학 논문의 중요도는 그것을 인용하고 있는 논문의 수 또는 저널의 영향력으로부터 평가되고 있다.[23] 그렇다면 웹 페이지의 중요도를 평가할 때에도 논문의 중요도가 평가되는 기준을 참조하면 되지 않겠는가? 즉, 중요도가 높은 웹 페이지는 다른 웹 페이지로부터 더 많이 링크된 것이라고 생각할 수 있다.

하지만 단지 하이퍼링크가 많이 이루어진 문서[24]라고 해서 곧바로 중요한 문서라고 단정하기 어렵다. 중요한 문서가 다른 문서를 링크하는 경우와 그렇지 않은 경우를 구별할 필요가 있다. 그래서 페이지랭크 알고리즘은 연결의 시작점에 해당하는 웹 페이지의 중요도까지 함께 고려한다. 결과적으로 페이지랭크 알고리즘은 각 웹 페이지마다 페이지랭크 값을 부여한다. 이 값은 ① 웹 페이지로 연결된 하이퍼링크를 포함하는 문서의 페이지랭크 값과 ② 하이퍼링크를 클릭할 가능성을 곱하여 계산된다.

페이지랭크 알고리즘의 작동 원리는 다음과 같다. 어떤 주제를 다룬 여러 웹 페이지들을 조사하는 사람이 있다고 생각해보자. 그는 임의로 선택된 웹 페이지부터 읽기 시작한다. 그리고 지루해지기 전까지는 그 웹 페이지로부터 하이퍼링크로 연결된 다른 웹 페이지로 임의로 이동해 가며 해당 주제에 관해 공부한다. 그리고 지루함을 느끼는 순간 이동을 종료한다. 그가 호기심이 많다면 지루함을 느끼게 될 가능성은 적고, 그렇지 않다면 지루함을 느끼게 될 가능성은 높을 것이다. 이를 감쇠 계수(damping factor)라고 한다. 감쇠 계수가 1에 가까울수록 이동이 증가하여 공부량도 늘어나게 된다. 브린과 페이지는 이 계수가 0.85이라고 가정하였다.

이러한 모형(random surfer)에서 페이지랭크 값은 임의로 웹 페이지를 탐색하는 사람이 특정한 웹 페이지에 방문할 가능성을 의미한다. 이를 이해하는 데에 도움이 될 만한 예시를 생각해보자. 하이퍼링크로 연결된 네 개의 웹 페이지가 존재한다고 가정

---

23) E. Garfield, "Citation Analysis as a Tool in Journal Evaluation: Journals can be ranked by frequency and impact of citations for science policy studies," 178 Science 471-479 (1972).

24)이처럼 하이퍼링크를 따라 웹 페이지로 들어오는 연결 고리의 수를 'in-degree'라 한다.

하자(그림 4).[25] 이 사례에서 어떤 방문자가 특정한 웹 페이지를 방문할 가능성은 일응 하이퍼링크의 개수를 기준으로 계산할 수 있다. 위 사례에서는 전체 하이퍼링크는 8개이고, 그 중 A 문서로 들어오는 하이퍼링크는 2개이므로, A 문서를 방문할 가능성은 0.25(=2/8)이다. 마찬가지의 방법으로 B의 가능성은 0.375(=3/8), C의 가능성은 0.25(=2/8), D의 가능성은 0.125(=1/8)가 된다.

하지만 이는 인용하는 문서가 얼마나 중요한 것인지를 고려하지 않은 결과이다. 페이지랭크 알고리즘은 더 중요한 문서로부터 인용된 문서가 더 높은 값을 얻도록 설계되어 있다. 즉, A의 중요도는 B의 중요도의 근거가 된다. 이는 하이퍼링크의 연결 구조에 따라 반영된다. 그리고 B의 중요도는 다시 A의 중요도의 근거가 된다. 이처럼 상호 인용이 이루어진 하이퍼링크가 존재하는 경우 페이지랭크 값을 곧바로 계산하기는 어렵다. 그러므로 일반적으로 페이지랭크는 반복적으로 값을 업데이트하는 반복 알고리즘(iterative algorithm)에 의하여 계산된다.

이러한 방법으로 계산하면 <표 1>과 같은 페이지랭크 값이 도출된다. 이 결과는 단지 하이퍼링크의 수만을 고려한 경우와는 꽤나 차이가 있다. 특히 중요도가 비교적 높은 A 문서와 B 문서가 함께 링크하고 있는 C 문서에 높은 페이지랭크 값이 부여된다는 점이 흥미롭다.

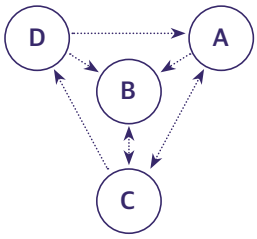

그림 4. 웹 페이지 간 연결 구조

	페이지랭크 값	순위
A 문서	0.258	3
B 문서	0.290	2
C 문서	0.387	1
D 문서	0.129	4

표 1 웹 페이지의 중요도와 순위

요컨대, 페이지랭크 알고리즘은 더 많은 웹 페이지로부터 링크가 이루어진 문서에 대해 높은 점수를 부여한다는 생각을 기본으로 하되, 중요도가 높은 문서로부터 이루어진 링크에 더 높은 점수를 부여하는 방식으로 웹페이지의 순위를 결정하는 알고리즘이라고 이해될 수 있다.

3. 다양한 순위 결정 알고리즘의 혼합 적용

페이지랭크 알고리즘에 따라 결정된 페이지랭크 값은 웹페이지의 중요도를 얼마나 잘 반영하는 것일까? 구글의 검색 서비스가 거둔 놀라운 상업적 성공을 고려한다면 이를 비교적 잘 반영하고 있다고 생각할 수도 있다. 하지만, 웹 페이지의 중요도를 그저 하이퍼링크만을 고려해서 도출해 낼 수 있을 것으로 기대하기란 어렵다.

그래서 구글도 페이지랭크 알고리즘을 유일한 측도로 사용하는 것은 아니다. 구글

25)Christopher Brinton and Mung Chiang, “PageRank by Google”, Networks Illustrated: Principles without Calculus, (2020).

은 자신의 검색 엔진이 "하나가 아닌 여러 개의 알고리즘으로 구성되어 있[고]", "검색어의 단어, 페이지의 관련성 및 유용성, 출처의 전문성, 사용자의 위치 및 설정과 같은 다양한 요소를 고려[한다.]"고 한다. 또한 "각 요소에 적용되는 가중치는 검색어의 성격에 따라 달라[진다.]"고 한다. 구체적으로 보면, 검색어의 의미를 이해하고 분석하는 데 사용되는 언어 모델이 있고, 그 의미와 일치하는 정보가 포함되어 있을 가능성을 통계적으로 판단하는 예측 모델이 있다. 사용자의 국가와 위치, 검색 및 계정에서의 활동 정보로부터 개인화된 모델도 있다.[26)]

구글뿐만 아니라 다른 검색 서비스도 마찬가지이다. 네이버도 그 검색 순위를 결정하는 데 있어 관련도, 즉 문서와 검색어 간의 유사성, 문서 자체의 품질, 이용자 선호도 등을 함께 고려한다고 한다. 유사성은 이용자가 입력한 검색어가 문서 내에 얼마나 잦은 빈도로 발견되는지, 문서 내에 어떤 위치에서 발견되는지 등을 가리키는 것이고, 문서 품질은 해당 문서에 걸린 링크의 수, 다양한 형태의 정보 포함 여부, 문서를 포함한 사이트의 업데이트 정도 등을 가리키는 것이다. 한편 이용자 선호도는 해당 문서에 얼마나 많은 이용자가 방문했는지, 이용자가 해당 문서에 얼마나 오래 머물렀는지 등을 가리키는 것이다. 한편 네이버는 'C-Rank'라는 알고리즘을 이용하여 해당 문서가 포함된 블로그의 신뢰도 또한 고려한다고 한다.[27)]

V. 결론

이상에서 알고리즘이 기술적 관점에서 어떠한 의미가 있는지 구체적인 사례를 통해 살펴보았다. 알고리즘이란 어떤 문제를 해결하기 위해 입력과 출력과의 관계를 정해 놓은 계산 절차로 이해할 수 있다. 이때 어떤 문제를 알고리즘 방식으로 해결하기 위해서는 주어진 문제를 명시적으로 기술된 계산 문제로도 환원해야 한다.

나아가 알고리즘을 효율성과 정확성이라는 기준을 통해 평가할 수 있다는 점도 살펴보았다. 알고리즘의 효율성을 평가하기 위해서는 흔히 '시간 복잡도'라는 개념을 사용하는데, 동일한 과제를 해결하는 알고리즘이라고 하더라도 (특히 처리할 자료량이 방대한 경우) 수행 시간에 있어 큰 차이가 발생할 수 있다는 점도 보았다. 또한 알고리즘의 정확성은 사양을 기준으로 하여 평가할 수 있는데, 이 경우 알고리즘이 부정확한 경우는 알고리즘이 사양을 충족시키지 못하는 경우와 사양 자체가 잘못 설계된 경우로 구분할 수 있었다. 특히 후자에 있어서는 사양을 설계하는 과정에서 규범적 판단이 (의식적으로나 무의식적으로) 포함되기 마련이므로, 사양의 설계 과정에서 알고리즘의 사회적 영향에 대한 고려가 이루어질 필요가 있다는 주장도 함께 검토하였다.

마지막으로 구글의 검색 알고리즘으로 잘 알려진 페이지랭크 알고리즘의 원리를 보았다. 페이지랭크 알고리즘은 하이퍼링크를 기준으로 웹 페이지에 점수를 부여하는 방식인데, 이는 검색 엔진이 고려하는 여러 요소 중 하나일 뿐이다. 구글을 포함하여 현재 활용되고 있는 검색 엔진은 그 이외에도 다양한 요소들을 고려하고 있다.

---

26) Google, "검색의 원리", Google 검색, https://www.google.com/intl/ko/search/howsearchworks/algorithms/ (2020. 9. 6. 확인).

27)네이버, "통합검색 노출기준", Naver 고객센터, https://help.naver.com/support/service/main.help?serviceNo=606&categoryNo=19642 (2020. 9. 6. 확인).